THE UNIVERSITY OF CHICAGO

MAKING INTERACTIONS WITH HOME IOT DEVICES MORE SECURE, PRIVATE, AND USABLE

A DISSERTATION SUBMITTED TO
THE FACULTY OF THE DIVISION OF THE PHYSICAL SCIENCE
IN CANDIDACY FOR THE DEGREE OF
PHILOSOPHY OF DOCTOR

DEPARTMENT OF COMPUTER SCIENCE

BY

WEIJIA HE

CHICAGO, ILLINOIS

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

The Internet of Things for homes (*home IoT*) creates unique security challenges. Home IoT devices often interact with multiple people under the same roof and are equipped with various modalities. They do not only react to commands from the user, but also from the environment, which increases the attack surface and changes the threat model. The highly fragmented ecosystem of home IoT devices only makes things worse, making it harder to find a solution that fits all devices.

Traditional security approaches fail in these challenges because they are designed for conventional computing devices like computers or smartphones, which are mostly used by one user with proper screens and keyboards. These characteristics make mechanisms like access control and authentication much more manageable. On the other hand, traditional computing devices are general-purpose, making enforcing allowlists of network traffic impossible. This is no longer the case for home IoT devices, and new strategies must be employed.

Responding to these emerging challenges in the home IoT, we create a road map about how to make a home IoT system secure and usable on different levels. We are mainly interested in devices' interactions with the external world, such as users, the physical environment, sensors, and remote servers. With such emphasis, we divide a home IoT system into three parts: user & software, environment & hardware, and network. For the user & software part, this thesis elicits requirements for access control systems that handle users with complex relationships and constantly changing contexts. For environments & hardware, this thesis creates a framework for context sensing, systematizing contexts and their required sensors, along with their security, privacy, and usability characteristics. In the network part, this thesis maps the design space of creating network allowlists that successfully generalize.

CHAPTER 1
INTRODUCTION

The proliferation of home IoT devices in recent years has raised significant security and privacy concerns [1]. Unlike traditional computing devices, such as computers or smartphones, home IoT devices often need to support multiple users with complex social relationships, as well as to react to multiple modalities [2, 3]. Both requirements create a larger attack surface than before. Moreover, many IoT devices are not designed for general-purpose computing. Devices intended for specific, narrow purposes inherently have different properties. Even for different home IoT devices made for the same purpose (e.g., different IoT thermostats), different manufacturers create different network architectures, causing more fragmentation in the home IoT ecosystem [4]. The widely existing variations in home IoT devices make protecting the security of all home IoT devices in the wild a challenge [5, 6, 7].

Unfortunately, many security practices and mindsets fail to recognize differences between home IoT devices and traditional general-purpose computing devices. Therefore, they fail to account for these differences. Access control in smart homes, for example, retains the admin-guests model that is typical on general-purpose computers even though the social relationships between IoT users are much more complicated [8, 9, 10, 11, 12, 13, 14]. IoT systems are also sensitive to environmental changes. Attackers can alter the system's behaviors by changing the environment [15, 16, 17, 18, 19, 20]. Security and privacy research for home IoT devices is often dedicated only to one type of devices [21, 22] or one particular home IoT platform [23, 24].

This thesis revisits and inspects various aspects of home IoT systems from a security and privacy perspective. The thesis rethinks old security mindsets transferred naively from general-purpose computing devices, pinpointing how they fail in the face of the unique challenges brought by home IoT. The thesis subsequently propose more adaptive, secure, and privacy-respectful designs. The thesis emphasizes the interactions a home IoT device may have with the external world, such as humans, environments, and remote servers. However, security and privacy concerns caused by vulnerabilities from the system itself (e.g., exploitable firmware, insecure API implementation) are out of the scope of this thesis because such threats are usually not unique to home IoT systems and have been studied extensively [7].

As with any other system, home IoT systems consist of various connected components, namely the following: *software*, *hardware*, and *network*. Different components enable distinct interactions with various audiences. For instance, software enables interactions between users and the system. Through software, a user can control devices, set up routines, or grant/revoke others' access to certain capabilities. Hardware, on the other hand, is the communication channel between the environment and the system. Sensors, the devices that detect changes in the environment, provide information to the system. Actuators, the devices that execute commands, perform actions that may change the environment. Similarly, a network connects the home IoT devices with each other and with remote servers, facilitating all functionalities the devices can provide. A more detailed system model is defined and illustrated in Chapter 3.

Each component in the home IoT system has its own goals. The audiences they are facing also may impact the system in various ways. These differences eventually create unique sets of challenges for each respective component, requiring diverse techniques to understand the obstacles and varied perspectives.

For these components, this thesis makes the following contributions:

Users & Software: This thesis focuses on security and privacy. As such, access control is a primary concern. The access control system is meant to manage the relationship between human users and the devices that are under the system's control. However, the complex social relationships among the users in the same household necessitate highly contextual access control policies, challenging the system to better understand the situation before making any decisions. Traditional access control systems for general-purpose computing devices rarely face such challenges. Adopting traditional access control approaches without thoughtful adaption leaves users with limited ways to express their intent to the system. The lack of tangible interfaces on many home IoT devices makes things worse.

Therefore, we need to understand what access control system is desired or expected by home IoT users. To achieve this, we studied the pitfalls of these fundamental security mechanisms in the age of IoT, exploring their necessary characteristics and features through a 425-participant online survey. We discovered the delicacy of different social relationship among users, helping us understand how these relationships altered users' default attitudes toward access control policies regarding various capabilities that a home IoT system can offer. We also curated a list of contexts that can affect users' access control decisions, shedding light on what an adaptive system should know to respect the constantly changing contexts in a home [8].

Environment & Hardware: As mentioned previously, hardware in home IoT can be categorized into sensors and actuators. To support the rich contexts we identified in the previous study, sensors are indispensable. We thus next explore various desired contexts, identifying assorted sensors that can actually sense them. The list of potentially suitable sensors is long; the challenge becomes which one to adopt. There are many dimensions that can affect this decision. For security, ensuring the correctness of sensor readings is crucial, which by itself is a challenging problem. People also care about their privacy. Sensors that are invasive (e.g., cameras) are less ideal for privacy. Some sensor implementations require considerable efforts from users, making them less likely to be adopted. Moreover, these dimensions may even conflict with each other, creating complex trade-offs. To tackle the challenge, we developed a decision framework for sensor adoption in the context of access control. We proposed a novel threat model that involves internal attackers and established detailed criteria for security, privacy, and usability. We then consider each sensor's characteristics in an adversarial setting and rate them based on the criteria we developed. The framework thus helps future users navigate trade-offs based on their own needs [25].

Remote Servers & Network: Network threats have existed almost as long as the internet itself. Corresponding network defenses like firewalls are well-established. However, the distributed nature of the home IoT makes the deployment of such defenses more challenging. Devices in the same home IoT system can be made by various vendors, who adopt different infrastructures that can affect the way these devices communicate with remote servers. The complexity of today's Internet also makes various devices (instances) of the same product behave differently, either due to load balancing adopted by the servers or the geo-location of the local device. A set of rules that permits network activities thus can vary from device to device, even if the devices are viewed as the same product in users' eyes. Therefore, creating generalizable allowlists is both desired and challenging.

Our work explores the design space of automatically creating generalizable allowlists for various home IoT products. We automatically create allowlists that can work for all devices of a product, instead of just one device, by using the IoT Inspector dataset, a large-scale IoT traffic dataset collected from real homes [26]. Leveraging this large-scale dataset, we discover that the network activities of different devices (of the same product) can greatly vary due to load balancing, under-represented regional services, user-specific settings, and more. These discoveries guide us to explore and map the design space needed to support generalizable allowlists. For instance, we consider various host representations (e.g., IP, hostname, domain), samples from different regions, sample sizes, and minimum bars for admitting a new host into the allowlist. We then measure how different allowlists can affect device functionality. Our discoveries on the general trends provide guidance on the desirable design for allowlists, such as the usage of hostname patterns, region-specific allowlists, and more. We also learned that certain infrastructures and careless host naming can lead to the failure of allowlists. We also deployed these allowlist on real-world devices in our lab to understand their impact on functionality. Given the prevalence of load balancing in today's networks, we found that hostname-pattern-based allowlists help to maintain devices' functionality, but still can fall short for video and music streaming devices. We also found allowlists to be relatively stable over time, in contrast to many people's belief.

Throughout this thesis, we create a road map that can secure home IoT devices from various attackers in different parts of home IoT systems. A smart home owner could use our context sensing framework to select a combination of home IoT sensors that not only fit their needs, but also are robust against local attackers while preserving user privacy. Once the selected home IoT devices and sensors are deployed, the owner can use allowlists to prevent attacks launched by a remote attacker or a compromised device. Finally, for legitimate users, a good access control system will assign default access based on their social roles in the household, as suggested by our work.

CHAPTER 2

RELATED WORK

2.1 Access Control in the Home IoT

Current research focuses on analyzing and fixing the security of platforms [27, 28, 29], protocols [30], and devices [31]. Fernandes et al. discuss how smart-home apps can be overprivileged in terms of their access to devices and present attacks exploiting deficiencies in apps' access-control mechanisms [27]. Mitigations have involved rethinking permission granting [32, 33, 27].

Comparatively little work has focused on authorizing and authenticating *humans to home IoT devices*.

Prior work has focused on the difficulties of access control in the home [34, 35, 36, 37], rather than solutions. Furthermore, the consumer device landscape has changed rapidly in the years since these initial studies.

Some older work has examined authentication [38] and access-control [39] for deployed home IoT devices, finding such affordances highly ineffective. Recent studies [2, 40] have sought to elicit users' broad security and privacy concerns with IoT environments, particularly noting multi-user complexity as a key security challenge. This complexity stems from the social ties in a home IoT setting. For instance, researchers have noted that roommates [12], guests [41], neighbors [13], and children [9, 42] are all important considerations in multi-user environments. In Chapter 4, we build on this work, identifying desired access-control rules for home IoT devices and bringing both relationships between home occupants and devices' individual capabilities to the forefront.

To take contextual factors into consideration, Schuster et al. proposed protocols for enforcing contextual (a.k.a. "situational") access control [43]. They introduced Environmental Situation Oracles (ESO) that answer queries about context. They did not, however, investigate physical sensing, which inspired our work in Chapter 5. In Chapter 5, we investigate the trustworthiness and usability of the physical sensing that necessarily underpins their oracles.

To improve the robustness of sensing, Birnbach et al. proposed ensemble methods that combine sensors to verify physical events in homes [44]. While they focused on techniques for sensor fusion, we provide a framework to help designers choose a set of sensors with complementary usability/privacy properties and abilities to resist attacks.

User-centered work focuses on the expression of policies, not enforcement. He et al. mapped potentially desirable policies [8] and Zeng et al. studied the user interface for expressing policies in multi-user homes [11], but gaps remain. Current designs rely on the integrity and availability of sensor data, which we show are not guaranteed. Privacy concerns can also make people unwilling to deploy certain sensors in homes [45].

Prior research on IoT authentication has focused on protocols (e.g., Kerberos-like frameworks [46, 47]) without considering the constraints of users. Feng et al. introduced VAuth, voice-based authentication for voice assistants [48]. VAuth requires the use of wearable hardware to establish an authentication channel, however.

Smartphones can be considered a predecessor to the IoT, yet the large literature [49, 50, 51, 52] on specifying which apps can access which resources translates only partially to home IoT devices. Enck et al. discuss how apps could gain access to resources by requesting permission from the user [51], while Felt et al. discuss how users may not always pay attention to such prompts [49]. A common theme is that apps access phone resources, and a phone is a single-user device not typically shared with others. On current versions of Android, one can configure secondary accounts with restrictions on what apps may be used [53], yet having separate accounts does not solve the multi-user challenges of home IoT devices.

2.2 Privacy in the Home IoT

Prior research demonstrates gaps between users' perceptions of their privacy in smart homes and reality. Users have a limited understanding of privacy risks, and these opinions vary by context [2, 54, 55, 40, 56]. Researchers have explored the design space of usable privacy protection for the IoT [57], designed mechanisms for data transparency in homes [58, 59, 60], and made personalized privacy mechanisms that predict an individual's preference in a given situation [61, 40]. Building on the usable privacy literature, we capture privacy concerns about implementations of contextual access control, including sensing, data storage, and retention.

There is a growing body of research on developing privacy-preserving measures in ubiquitous sensing systems, such as obfuscation in audio sensing [62], cross-device tracking through ultrasound [63], bystander privacy in wearable cameras [64, 65], and so on. However, the public has no guarantee that the manufacturers of smart home devices will implement any of these counter-measures. Non-technical users are also unlikely to deploy such measures by themselves. Therefore, smart home designers should select a sensing method that minimizes overprivileged and inadvertent data collection before deployment, which is the focus of this thesis.

2.3 Systematization of Knowledge (SoK) in IoT

A few prior papers survey specific aspects of the IoT. Fernandes et al. highlighted that access control and authentication are among the IoT's new intellectual challenges [66]. Considering software and networks, Alrawi et al. proposed methods for security evaluations of home IoT devices [7]. While they comprehensively explore digital attacks, we instead focus on physical attacks on sensors. Yan et al. examined analog sensor security, formalizing sensor circuits' security properties [67]. Their attackers are highly technical, whereas we focus on the non-technical adversaries that are common inside homes. On the network level, Yu et al. argued that context-aware enforcement is essential in the IoT [6]. Zhang et al. [5] compared academic and industry perspectives on IoT security. Their “environment mistrust” category can include physical attacks on sensing. They focus on technical attacks, such as signal jamming and voice synthesis. We expand their threat model to explore physical attacks on sensors by non-experts.

2.4 Generating IoT Network Policies

Chapter 6 aims in part to understand how allowlists generalize across devices for various home IoT products. Creating an allowlist 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 [68]. Prior research on IoT network blocklists takes a more advanced approach, creating broadly applicable blocklists by blocking observed hostnames one at a time on a real IoT device [69]. Because blocklists do not have to be complete to maintain device functionality, blocklists can be deployed immediately.

Creating generalizable allowlists is a different story. Although several studies proposed creating allowlists based on observing a single device [70, 71, 72, 73], our analysis shows that due to load balancing and other network-specific settings, these allowlists will not generalize. Rather than focusing on a single device, we used the IoT Inspector dataset, the largest home IoT traffic dataset, to understand whether an allowlist can transfer to other devices. Our work also contrasts with proposals like Cisco’s Manufacturer Usage Descriptions (MUD) [74], which asks IoT manufacturers to specify Internet hosts with which their IoT device should communicate. Specifically, our method requires no additional work from IoT vendors.

Thomasset et al. proposed SERENIoT, which uses blockchains to create and maintain allowlists from multiple devices [75]. While SERENIoT also creates allowlists through a data-driven approach, storing the data on a blockchain, it does not study the process of generalizing this traffic data, nor evaluate how policies generalize across devices. It is also unclear how the tool performs outside the lab, where variations like load balancing make consensus hard to reach. In contrast, we evaluate allowlist designs through both a retrospective simulation on thousands of devices and a lab study.

Prior work has also created network policies based on application behaviors like heartbeats and firmware checks [76]. However, doing so requires knowledge about application data features, which are not always collected for privacy reasons. Our method also contrasts with systems like Bro [77] that focus on analyzing network flows at high speeds, forcing the user to decide what policies to implement.

2.5 Data-Driven Approaches for Anomaly Detection

To evaluate allowlist generalization in Chapter 6, we use the IoT Inspector dataset [26]. Our use of a large-scale, real-world dataset also differs from prior work. Many existing approaches use supervised machine learning techniques to detect anomalous flows [78, 79, 80, 81, 82, 83, 84, 6, 85], but they typically rely on training and evaluation data from dozens of devices in a lab environment. For example, Dong et al. [86] trained neural networks to fingerprint IoT traffic based on 10 IoT devices in the lab. Similarly, Mandalari et al. [69] identified blockable IoT-contacted hostnames based on an in-lab study of 31 devices. It is unclear whether these techniques would apply to a larger IoT traffic dataset. Some commercial products [87, 88] claim to generate rules using proprietary methods and data, but do not detail their approach.

In addition to the scale, our novelty also lies in the realism of the dataset. The dataset was collected from 5,439 volunteers from around the world [26], reflecting devices deployed under real-world conditions. Much existing work, such as from DeMarinis et al. [89] and even the MUD specification [74], assumes that IoT devices have predictable patterns in network traffic. Also, prior work, including from Dong et al. [86] and Mandalari et al. [69], is based on static snapshots of IoT device behaviors in the lab, missing dynamic changes from new interactions and firmware updates.

IoT devices' potentially variable and complex behaviors call for techniques that dynamically build firewall-style rules. Other dynamic techniques include HANZO, which identifies devices and classifies network traffic from the home gateway [90]. Like much existing work, HANZO was evaluated in a lab setting, and its capability to generalize to the complex behaviors of real-world devices was not studied. HanGuard instead constructs network policies based on the traffic between IoT devices and mobile apps [91]. Our method does not require analysis of companion mobile apps.

2.6 Alternate Approaches to Network Security

Other work tries to protect compromised IoT devices using very different approaches to network security. For example, Martin et al. protect IoT devices on private networks from external attacks by scrambling port forwarding on the gateway [92]. Acar et al. propose securing HTTP servers on IoT devices to prevent the lateral movement of malware [93]. To reduce the attack surface, Goutam et al. divide IoT devices into controllable (e.g., lights) and non-controllable (e.g., hubs) devices, forcing the former to communicate only with the latter [94]. We instead examine communication between home IoT devices and Internet hosts.

CHAPTER 3
MODELING HOME IOT INTERACTIONS AND THREATS

In this chapter, we introduce our system model for a home IoT environment, with a focus on the interactions that may occur. Based on the defined model, we further illustrate the potential threats that can be introduced to the model.

Home IoT systems are inherently complex. The variety of control the system provides to its users is unseen before. To turn on a light in a home IoT system, for example, one can flip a physical switch, tap a button on their smartphone, ask the voice assistant to do it for them, or program the system so that the light turns on whenever the system detects someone is in the room.

Although the goal of adding more modalities to home IoT is to make people's lives easier, integrating everything without careful design can cause more problems than benefits, introducing security, privacy, and usability issues. By adding sensors to the system and allow them to provoke actions, attackers can now take advantage of the system via both network and sensing channels, compromising or deceiving the system for their own benefits [95, 19, 96, 97, 25]. Users also struggle to configure the system in a way that matches their expectations, either due to a misunderstanding about the system design or the system's inability.

Comprehensively understanding the potential security, privacy, and usability pitfalls in the interactions thus requires defining a home IoT system model. With a home IoT system model, we can identify the possible interactions that can be a point of access.

The home IoT system model we use in this thesis includes three main parts: software, hardware, and network, as showcasing in Figure 3.1. Each interacts with different stakeholders in a home IoT environment, triggering various security, privacy, and usability concerns and requiring distinct analysis and defense.

In the following parts, we detail the definition of our system model and its interactions with possible stakeholders, specifying the problems the system faces for each part.

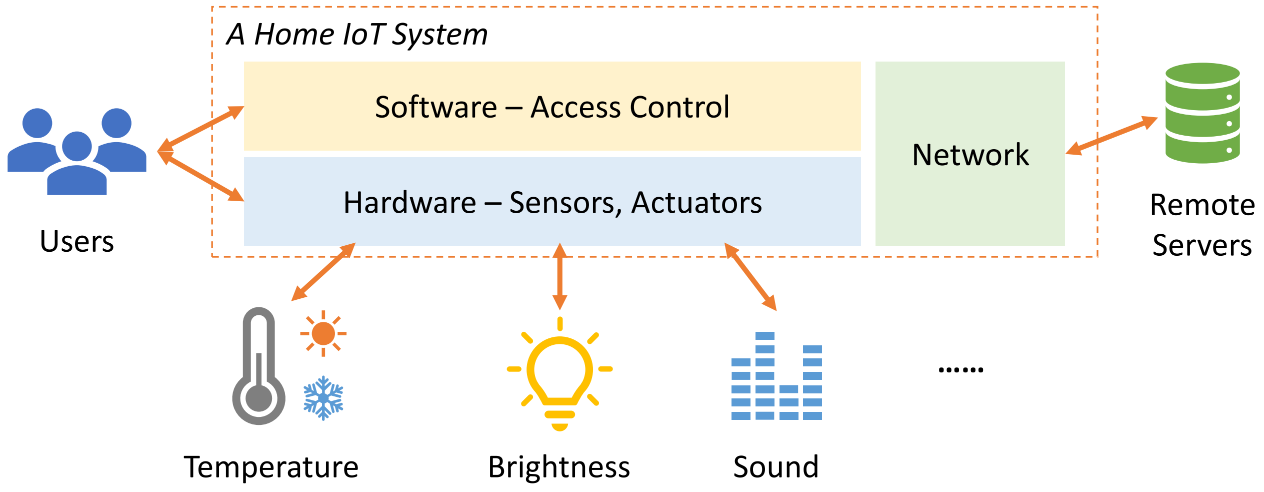


Figure 3.1: The home IoT stack model used in this thesis. It emphasize the interactions a home IoT may have with the external world, namely users, surroundings, and remote servers.

3.1 General System Setting

Although our system model includes three major parts, all parts share the same system model. In this section, we present the home IoT system model that we use throughout the thesis, specifying fundamental definitions of various stakeholders and the interactions methods they leverage.

Users Anyone with legitimate access to the system would be considered as a valid user, regardless of their residence in the house hold. Family members, visitors, and domestic workers can all be valid users in our system model.

Home IoT Devices Home IoT devices are Internet-connected devices. Our system model consists of two types of home IoT devices: *actuators* that execute commands (e.g., lights), and *sensors* that measure their surroundings (e.g., motion sensors). Once connected to the Internet, actuators can be controlled either locally (e.g., through physical buttons, or smartphones under the same network) or remotely (e.g., through commands sent from the Internet). Sensors usually cannot be explicitly controlled, but they can collect data from the environment and report the data either to a local hub or a remote server, informing the system about the changes of the physical attributes in the surroundings.

Interaction Modalities Interactions modalities describe how the user interacts with home IoT devices. We assume a domestic setting where occupants control home IoT devices through six different modalities, which includes smartphones apps, manual interaction, physical tokens, voice assistants, gestures, and automations. For example, a maintenance worker may unlock the front door using a smartphone app, while a child might turn off their lights by speaking to a voice assistant.

1. Smartphone Apps: Smartphone apps are by far the most common controlling method for home IoT. The smartphone app can be a home IoT device's companion app. It can also be some general management app that controls all the devices in one's home.

2. Manual Interaction: A user can interact with devices manually, often by flipping switches or pressing buttons. Additional sensing is required to identify the user in such scenarios. A contextual access-control framework can inform a smart device whether to permit access.

3. Physical tokens: Similar to using keys to unlock traditional doors, some home IoT device may also use some physical object as one way to recognize the user. It can be a token specifically made for authentication purpose. It can also be some daily used object with communication methods built in, such as smartphones. One example could be Apple Homepod's handoff feature [98]. Users can let a Homepod play songs from their phones by simply putting the smartphone nearby. Because users already authenticate to their phone, current IoT systems can rely on the possession of a phone as a proxy for identity.

4. Voice: Voice assistants let users control devices by speaking. Currently, they perform no authentication [17] or use speaker recognition that is easy to fool [99, 100].

5. Gestures: Currently uncommon in homes, gestures could be detected using ultrasonic or radio waves to recognize and authenticate movements as a source of input.

6. Automation: Smart home automation can link changes in context or other triggers to actions. They can be set with apps [32] or end-user programming [101]. Absent access control, automations may create loopholes [102, 23]. Imagine the automation: “If the lights turn off then play a movie.” If a child may not play movies, yet may turn off lights, a crafty child could start a movie by turning off a light. While focused on contextual access control, our framework can also apply to automations triggered by a sensed context [101], such as when a room is warm [103, 23]. An attacker who tricks a sensor can cause chained automations toward a malicious goal [102, 29].

Network Settings We assume that all the home IoT devices connected to a router to access the Internet. The router has the ability to interfere with the network behaviors of connected devices and is completely under users' control. There may be local communications between devices through WiFi, Bluetooth, Zigbee, etc. However, this thesis focus more on the interactions that happens between the system with external parties (e.g., remote servers) rather than the internal interactions.

3.2 User & Software

In Chapter 4, we discuss about how software (access control specifically) should be designed for home IoT users in depth.

Using software (e.g., smartphone apps) is still the primary way users interact with the home IoT system. Aligning the software design and users' expectations is crucial but often ignored. Neglecting users' expectations can lead to frustrations and misunderstandings, causing usability and security issues. In Chapter 4, we study users' expectations and perceptions about access control in a home IoT environment, identifying factors not covered by today's home IoT system and the design we should apply in the future.

In this thesis, *software* includes any software that controls the home IoT devices' behaviors, such as access control, authentication, home automation apps, etc. However, since the focus of this thesis is about security and privacy and authentication is naturally covered by access control, we will focus on access control in the following parts.

We assume a central access control system that can grant or revoke anyone's access to any home IoT devices. Devices are managed through a hub that facilitates communication between devices, enforces policies, and often allows for the creation of end-user programs or the use of apps.

Users are the primary audience of the software, as defined in Section 3.1. To better understand the unique challenges presented in a home setting, we categorize different users based on their relationships with the owner (i.e., the user that sets access control policies), such as spouse, children at different age, visiting family members, domestic workers, etc.

3.3 Environment & Hardware

Chapter 5 focuses on creating a framework for a contextual access control system. It heavily relies on the interactions between the environmental changes and the hardware used to detect these changes and act upon them.

A contextual systems means that the system can take different actions based on various contexts. For a contextual access control system, it means distinct access control policies can be activated based on the detected contexts, even for the same user.

Contexts describe a particular state of the physical world. In a smart home, contexts describe situations, states of actuators, presence of specific people, and more. Examples include a security camera being activated, the temperature staying within some range, or a specific person sitting in the kitchen. Contextual access control relies on sensors to reconstruct these situations, which we call *context sensing*. Once the context is sensed, contextual access control can then enforces policies through actuators by responding or not responding to a users' access request to a particular capability.

3.4 Remote Servers & Network

In Chapter 6, we detail the possible threats from the network and discuss how much security guarantee that a security mechanism like an allowlist can provide without hurting the devices' functionality. The chapter heavily relies on our assumptions of a home IoT user's network settings.

As defined in Section 3.1, the network part of our system model only includes one single local network, to which every device connects. The users have complete control over what these devices can send to or receive from the Internet. However, we do not assume the users would perform any deep package inspection on the traffic, given that most traffic is encrypted.

The remote servers include any servers that a device may contact during usage, including both first-party and third-party servers, regardless of whether they are essential or not.

CHAPTER 4

RETHINKING ACCESS CONTROL FOR THE HOME IOT

Recent years have seen a proliferation of Internet of Things (IoT) devices intended for consumers' homes, including Samsung SmartThings [104], the Amazon Echo voice assistant [105], the Nest Thermostat [106], Belkin's Wemo devices [107], and Philips Hue lights [108]. To date, IoT security and privacy research has focused on such devices' insecure software-engineering practices [31, 109, 27], improper information flows [109, 23, 29], and the inherent difficulties of patching networked devices [5, 6].

Surprisingly little attention has been paid to *access-control-policy specification* (expressing which particular users, in which contexts, are permitted to access a resource) or *authentication* (verifying that users are who they claim to be) in the home IoT. This state of affairs is troubling because the characteristics that make the IoT distinct from prior computing domains necessitate a rethinking of access control and authentication. Traditional devices like computers, phones, tablets, and smart watches are generally used by only a single person. Therefore, once a user authenticates to their own device, minimal further access control is needed. These devices have screens and keyboards, so the process of authentication often involves passwords, PINs, fingerprint biometrics, or similar approaches [110].

Home IoT devices are fundamentally different. First, numerous users interact with a single home IoT device, such as a household's shared voice assistant or Internet-connected door lock. Widely deployed techniques for specifying access-control policies and authenticating users fall short when multiple users share a device [2]. Complicating matters, users in a household often have complex social relationships with each other, changing the threat model. For example, mischievous children [9], parents curious about what their teenagers are doing [10], and abusive romantic partners [111] are all localized threats amplified in home IoT environments.

Furthermore, few IoT devices have screens or keyboards [3], so users cannot just type a password. While users could possibly use their phone as a central authentication mechanism, this would lose IoT devices' hands-free convenience, while naïve solutions like speaking a password to a voice assistant are often insecure.

Real-world examples of the shortcomings of current access-control-policy specification and authentication for home IoT devices have begun to appear. A Burger King TV commercial triggered Google Home voice assistants to read Wikipedia pages about the Whopper [17], while the cartoon South Park mischievously triggered Amazon Echo voice assistants to fill viewers' Amazon shopping carts with risqué items [16]. While these examples were relatively harmless, one could imagine a rogue child remotely controlling the devices in a sibling's room to annoy them, a curious babysitter with temporary access to a home perusing a device's history of interactions, or an enterprising burglar asking a voice assistant through a cracked window to unlock the front door [15].

In this chapter, we take a first step toward rethinking the specification of access-control policies and authentication for the home IoT. We structure our investigation around four research questions, which we examine in a 425-participant user study. These research questions are motivated by our observation that many home IoT devices combine varied functionality in a single device. For example, a home hub or a voice assistant can perform tasks ranging from turning on the lights to controlling the door locks. Current access control and authentication is often based on a device-centric model where access is granted or denied per device. We move to a capability-centric model, where we define a capability as a particular action (e. g., ordering an item online) that can be performed on a particular device (e. g., a voice assistant). Intuition suggests that different capabilities have different sensitivities, leading to our first research question:

RQ1: Do desired access-control policies differ among capabilities of single home IoT devices? (Section 4.4.2 and 4.4.3).

We investigated this question by having each study participant specify their desired access-control policy for one of 22 home IoT capabilities we identified. For household members of six different relationships (e. g., spouse, child, babysitter), the participant specified when that person should be allowed to use that capability. Our findings validated our intuition that policies about capabilities, rather than devices, better capture users' preferences. Different capabilities for voice assistants and doors particularly elicited strikingly different policies.

While the ability to specify granularly who should be able to use which capabilities is necessary to capture users' policies, it incurs a steep usability cost. To minimize this burden through default policies, we asked:

RQ2: For which pairs of relationships (e. g., child) and capabilities (e. g., turn on lights) are desired access-control policies consistent across participants? These can be default settings (Section 4.4.4).

In our study, nearly all participants always wanted their spouses to be able to use capabilities other than log deletion at all times. Participants also wanted others to be able to control the lights and thermostat while at home. As intimated by the prior policy, the context in which a particular individual would use a capability may also matter. Children might be permitted to control lights, but perhaps not to turn the lights on and off hundreds of times in succession as children are wont to do. Nor should children be permitted to operate most household devices when they are away from home, particularly devices in siblings' rooms. A babysitter unlocking the door from inside the house has far fewer security implications than the babysitter setting a persistent rule to unlock the front door whenever anyone rings the doorbell.

RQ3: On what *contextual factors* (e. g., location) do access-control policies depend? (Section 4.4.5).

In addition to a user's location, we found that participants wanted to specify access-control policies based on a user's age, the location of a device, and other factors. Almost none of these contextual factors are supported by current devices. Finally, to identify promising directions for designing authentication mechanisms in the home IoT, we asked:

RQ4: What types of authentication methods balance convenience and security, holding the potential to successfully balance the consequences of falsely allowing and denying access? (Section 4.4.6).

Analyzing consequences participants noted for falsely allowing or denying access to capabilities, we identify a spectrum of methods that seem promising for authenticating users (Section 4.5), thereby enabling enforcement of users' desired access-control policies for the home IoT.

4.1 Background

In this section, we scope our threat model and review current devices' support for access control and authentication.

4.1.1 Threat Model

The two major classes of adversaries in the smart home are external third parties and those who have legitimate physical access to the home. The former class includes those who exploit software vulnerabilities in platforms [27], devices [31] (e. g., with Mirai), or protocols [30] intending to cause physical, financial, or privacy-related damage. The latter class includes household members with legitimate digital or physical access to the home, such as temporary workers or children [9]. These insider threats have received far less research attention, but are the focus of this chapter. Insiders might be motivated to subvert a smart-home system's access controls for reasons ranging from curiosity to willful disobedience (e.g., a child attempting to take actions forbidden by their parents), or to attempt to correct imbalances created by the introduction of devices whose surveillance implications grant asymmetric power to certain members of a household (e. g., a parent tracking a teenager [10]).

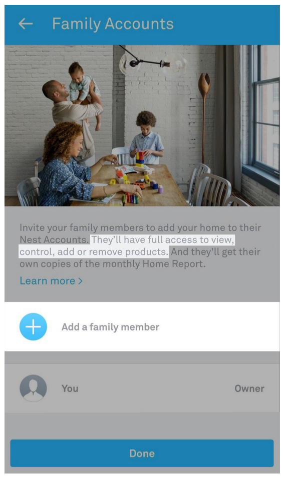
4.1.2 Affordances of Current Devices

Current home IoT devices have relatively limited affordances for access control and authentication. Taking a five-year-old survey of the home IoT landscape as a starting point [39], we surveyed current devices' affordances; Figure 4.1 shows representative samples. To control many current devices, people use smartphone apps that must be paired with devices. These apps offer various access-control settings. For example, the Nest Thermostat supports a binary model where additional users either have full or no access to all of the thermostat's capabilities. The August Smart Lock offers a similar model with guest and owner levels. Withings wireless scales let users create separate accounts and thus isolate their weight measurements from other users. On Apple HomeKit, one can invite additional users, restricting them to: (a) full control, (b) view-only control, (c) local or remote control.

Some devices offer slightly richer access-control-policy specification. The Kwikset Kevo Smart Lock allows access-control rules to be time-based; an owner can grant access to a secondary user for a limited amount of time. We find in our user study that time is a desirable contextual factor, but one of only many. We focus on capabilities, rather than devices. While most current devices do not allow for access-control policies that distinguish by capability, Samsung SmartThings lets users restrict third-party apps from accessing certain capabilities [112]. We find that restricting users, not just apps, access to a particular capability is necessary.

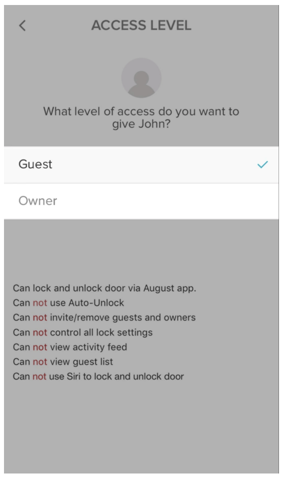
From this analysis, we found current mechanisms to be rudimentary and lack the necessary vocabulary for specifying access-control rules in complex, multi-user environments. We aim to establish a richer vocabulary.

Current authentication methods for the home IoT appear transplanted from smartphone and desktop paradigms. Passwords are widely used in conjunction with smartphones. For example, SmartThings has an app through which a user can control devices. A user first authenticates to this app using a password. Voice-based authentication is currently very


Family Accounts

Invite your family members to add your home to their Nest Accounts. They'll have full access to view, control, add or remove products. And they'll get their own copies of the monthly Home Report.

(a) Nest Learning Thermostat

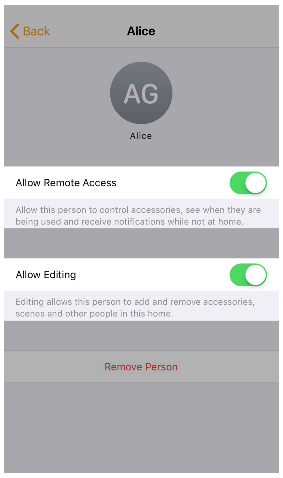

ACCESS LEVEL

What level of access do you want to give John?

Guest ✓
Owner

Can lock and unlock door via August app.
Can not use Auto-Unlock
Can not invite/remove guests and owners
Can not control all lock settings
Can not view activity feed
Can not view guest list
Can not use Siri to lock and unlock door

(b) August Smart Lock



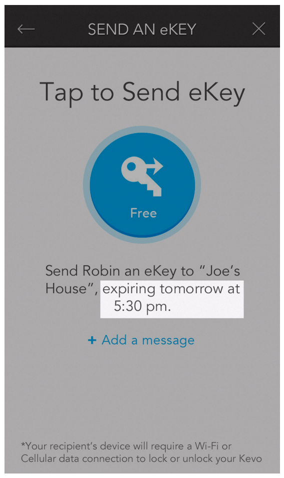
Allow Remote Access

Allow this person to control accessories, see when they are being used and receive notifications while not at home.

Allow Editing

Editing allows this person to add and remove accessories, scenes and other people in this home.

(c) Apple HomeKit


SEND AN eKEY

Tap to Send eKey

Send Robin an eKey to "Joe's House", expiring tomorrow at 5:30 pm.

*Your recipient's device will require a Wi-Fi or Cellular data connection to lock or unlock your Kevo

(d) Kwikset Kevo Smart Lock

Figure 4.1: Current access-control-specification interfaces: The Nest Thermostat (a) only allows "all-or-nothing" specification, while the August Smart Lock (b) only offers coarse-grained access control via predefined Guest and Owner groups. In contrast, Apple's HomeKit (c) differentiates between view and edit access level, as well as local and remote access. The Kwikset Kevo Smart Lock (d) provides time-based access control, but not other factors.

rudimentary and is not used for security, but for personalization. For instance, Google Home uses speaker recognition for customizing reminders, but not for security-related tasks [113].

4.2 Pre-Study

As a first step in exploring access control based on capabilities and relationships in the home IoT, we conducted a pre-study to identify capabilities and relationships that elicit representative or important user concerns. To ground our investigation of capabilities of the home IoT in devices consumers would likely encounter, we created a list of home IoT devices from consumer recommendations in CNET, PCMag, and Tom's Guide [114]. We grouped devices by their core functionality into categories including *smart-home hubs*, *door locks*, and *voice assistants*.

For each category of device, we collected the capabilities offered by currently marketed devices in that category. We added likely future capabilities, as well as the ability to write end-user programs [23, 29]. We showed each pre-study participant all capabilities identified for a single given class of device. The participant answered questions about the positive and negative consequences of using that capability, and they also identified additional capabilities they expected the device to have. We used this process to identify a comprehensive, yet diverse, set of capabilities that range from those that elicit substantial concerns to those that elicit none.

To identify a small set of relationships to investigate in the main study, we also showed participants a table of 24 relationships (e. g., teenage child, home health aide) and asked them to group these relationships into five ordered levels of desired access to smart-home devices. We chose this list of 24 relationships based on existing users and groups in discretionary access control (DAC) systems and common social relationships in households.

We conducted the pre-study with 31 participants on Amazon's Mechanical Turk. Participants identified potential concerns for a number of capabilities, in addition to identifying capabilities (e. g., turning on lights) that aroused few concerns. We used these results to generate a list of capabilities, grouping similar functionalities across devices into categories like viewing the current state of a device. We selected the 22 capabilities whose pre-study results showed a spectrum of opinions and concerns while maintaining a feature-set representative of smart homes.

To narrow our initial list of 24 relationships to a tractable number, we examined how pre-study participants assigned each relationship to one of the five ordered categories of desired access to household devices. We chose the six relationships that span the full range of desired access and for which participants were most consistent in their assignments to a category.

4.3 Main Study

To elicit desired access-control policies for the home IoT, our main study was an online survey-based user study. We recruited participants on Mechanical Turk, limiting the study to workers age 18+ who live in the United States and have an approval rating of at least 95 %.

4.3.1 Protocol

Each participant was presented with a single capability (e.g., "see which lights in the home are on or off") randomly chosen from among the 22 identified in the pre-study.

We then presented the participant with one of six relationships: spouse; teenage child; child in elementary school; visiting family member; babysitter; neighbor. We first asked whether such a person should be permitted to control that capability "always," "never," or "sometimes, depending on specific factors." These answers were the first step in identifying participants' desired access-control policies. For the first two options, we required a short free-text justification. To better understand the importance of an authentication method correctly identifying the person in question and the system correctly enforcing the access-control policy, we asked participants who answered "always" or "never" to state how much of an inconvenience it would be if the system incorrectly denied or allowed (respectively) that particular user access to that capability. Participants chose from "not an inconvenience," "minor inconvenience," or "major inconvenience," with a brief free-text justification.

If the participant chose "sometimes," we required additional explanations to further delineate their desired access-control policy. They first explained in free-text when that person should be allowed to use that capability, followed by when they should not be allowed to do so. On a five-point scale from "not important" to "extremely important," we asked how important it was for them to have (or not have) access to that capability.

We repeated these questions for the other five relationships in random order. Thus, each participant responded for all six relationships about a single capability.

Afterwards, we asked more general questions about specifying access-control policies for that capability. In particular, we presented eight contextual factors in randomized order, asking whether that factor should influence whether or not anyone should be permitted to use that capability. The possible responses were "yes," "no," and "not applicable," followed by a free-response justification. We asked about the following factors: the time of day; the location of the person relative to the device (e.g., in the same room); the age of the person; who else is currently at home; the cost of performing that action (e.g., cost of electricity or other monetary costs); the current state of the device; the location of the device in the home; the person's recent usage of the device. Further, we asked participants to list any additional factors that might affect their decision for that capability.

We concluded with questions about demographics, as well as the characteristics of the participant's physical house and members of their household. We also asked about their ownership and prior use of Internet-connected devices. We compensated participants \$ 3.50 for the study, which took approximately 20 minutes and was IRB-approved.

4.3.2 Analysis

Participants' responses about their access-control preferences included both qualitative free-text responses and multiple-choice responses. Two independent researchers coded the qualitative data. The first researcher performed open coding to develop a code book capturing the main themes, while the second coder independently used that same code book. To quantitatively compare multiple-choice responses across groups, we used the chi-squared test when all cell values were at least 5, and Fisher's Exact Test (FET) otherwise. For all tests, $\alpha = .05$, and we adjusted for multiple testing within each family of tests using Holm correction.

4.3.3 Limitations

The ecological validity and generalizability of our study are limited due to our convenience sample on Mechanical Turk. Most of our questions are based on hypothetical situations in which participants imagine the relationships and capabilities we proposed to them and self-report how they expect to react. Furthermore, while some participants were active users of home IoT devices, others were not, making the scenarios fully hypothetical for some participants. We chose to accept this limitation and include recruits regardless of prior experience with home IoT devices to avoid biasing the sample toward early adopters, who tend to be more affluent and tech-savvy.

4.4 Results

In the following sections we present our findings. We begin by providing an overview of our participants (Section 4.4.1). Next, we present how desired access-control policies differ across capabilities (RQ1, Section 4.4.2) and the degree to which desired policies differ across relationships (RQ1, Section 4.4.3). After that, we show for which pairs of relationships and capabilities the desired access-control policies are consistent across participants. We use these pairs to derive default policies (RQ2, Section 4.4.4). Next, we evaluate which contextual factors (e. g., age, location, usage) influence the “sometimes” cases the most, thus explaining users’ reasoning for not always allowing access to a capability (RQ3, Section 4.4.5). Finally, we analyze the consequences of false authorization and show the impact of falsely allowing / denying access to a certain capability on a per-relationship level (RQ4, Section 4.4.6).

4.4.1 Participants

A total of 426 individuals participated in the study, and 425 of them were qualified as effective responses. One response was excluded from our data because their free-text responses

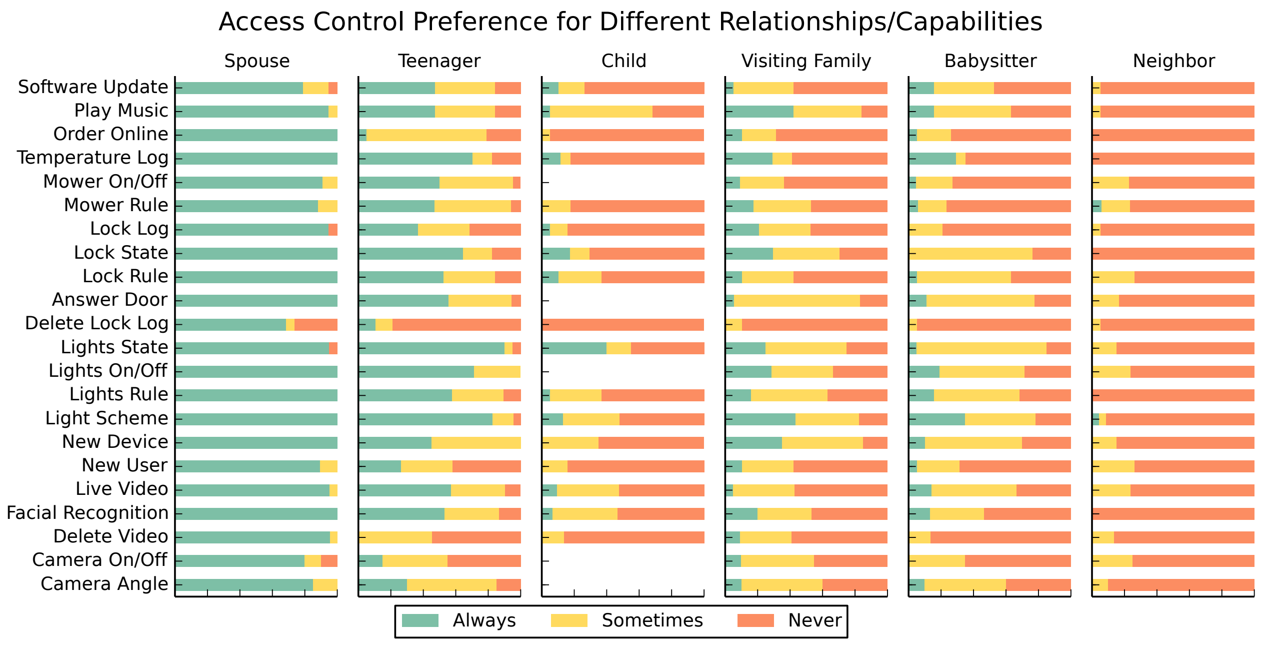


Figure 4.2: Participants' desired access-control policies. We introduced participants to a list of relationships (e.g., *neighbor*) and asked them to choose whether someone of that relationship should be permitted to "always," "sometimes," or "never" control a capability (e.g., adjust the *camera angle*) in their smart home.

were unrelated to our questions. Our sample was nearly gender-balanced; 46 % of participants identified as female, and 54 % as male. The median age range was 25-34 years old (47 %). Most participants (85 %) were between 25 and 54 years old. Some participants (19 %) reported majoring, earning a degree, or holding a job in computer science or a related field.

The majority of our participants (67 %) live in a single-family home, while 25 % live in an apartment. Nearly half of the participants own (49 %) the place where they live, while 47 % rent. Furthermore, we asked how many people (including the participant) live in the same household. Around 20 % of participants reported living in a single-person household, 27 % in a two-person, 23 % in a three-person, and 17 % in a four-person household.

4.4.2 Capabilities (RQ1)

Current access-control implementation in a smart home system is largely device-based. However, our data motivates a more fine-grained, flexible access-control mechanism. In the following parts, we discuss our main findings, which are visualized in Figure 4.2.

A) **Capability Differences Within a Single Device**

We observed that participants' attitudes toward various capabilities differ within a single device. For example, voice assistants can be used to play music and order things online. However, participants were much more willing to let others play music (32.5 % of participants choose *never* averaged across the six relationships, $\sigma = 0.33$, $median = 23.7\,\%$) than order things online (59.7 % choose *never* on average, $\sigma = 0.40$, $median = 71.1\,\%$) (FET, $p < .05$ for the teenager, child, and visiting family member relationships).

Another example of differing opinions across capabilities within a single device include deleting an IoT lock's activity logs and answering the door, viewing the current state of the lock, and setting rules for the lock. Across relationships, participants were permissive about capabilities like answering the door (25.6 % chose "never" averaged across all relationships other than children, $\sigma = 0.33$, $median = 16,7\,\%$). Because children would likely not have a smartphone, we did not ask about them performing this action and we exclude them from this analysis. In contrast, 76.8 % of participants said they would *never* allow others to delete activity logs ($\sigma = 0.28$, $median = 92.1\,\%$). These differences are significant (FET, all $p < 0.05$ comparing within teenagers, visiting family, and babysitters). Even for a very trust-based relationship like a spouse, some participants still chose *never.* When asked why, one participant wrote: *"No one should be able to delete the security logs."*

Even if individuals with relationships like neighbor or babysitter do not live in the same house, permissions are sometimes given when the owner of the house is not around. One typical response for when a capability should be accessible to neighbors is *"Perhaps when I'm on vacation and I ask them to watch my home."*

B) **Context-Dependent Capabilities**

We identified "Answering the Doorbell" to be a highly context-dependent capability. 40 % of participants across relationships ($\sigma = 0.33$, $median = 38.9\,\%$) selected *sometimes* for this capability. At the same time, an average of 25.6 % of participants across relationships chose *never* ($\sigma = 0.33$, $median = 16.7\,\%$).

Whether the homeowner is present is a key factor impacting responses. Many participants (66.7 %) chose *sometimes* when it came to the babysitter, because the job itself indicates the parents are not around. If a delivery person rings the doorbell while the babysitter is home, the babysitter should be allowed to handle the event. The majority of participants (77.8 %) also *sometimes* trust a visiting family member with the same level of access. Some participants (16.7 %) will even consider giving this access to their neighbors, so that if there is an emergency when the family is on vacation, their neighbor can see who is at the door from their smartphone.

4.4.3 Relationships (RQ1)

Relationships play an important role in participants' preferred access-control policies.

A) **Babysitter vs. Visiting Family**

In the pre-study, we identified the babysitter and a visiting family member to be members of a guest-like group. In the main study, participants' overall attitudes toward babysitters and visiting family members were quite consistent with each other. No significant differences are observed between these two relationships in our pairwise chi-squared tests. This is understandable because both relationships share some trust with the homeowner, while neither lives in the same household.

In general, policies toward a visiting family member are slightly more permissive than policies toward a babysitter. However, analyzing the qualitative data, we found the situation to be more complex. There are some specific capabilities, such as "Live Video," where babysitters would be granted permissions at a higher rate than a visiting family member. 57.1 % of participants decided that a visiting family member would *never* have access to this feature, while only 33.3 % of participants decided the same for a babysitter. The reason is that a babysitter's job is to take care of a child while a parental figure is away. Therefore, the capability itself might help a babysitter take better care of the child, leading to a high rate of granting this permission *sometimes*.

Meanwhile, some features show strong subjective variations, including granting babysitters and visiting family members permission for "Answering the Doorbell." Some participants found it useful to always allow access, while other participants felt uncomfortable letting someone that is not part of their family have access to this particular capability.

From these observations, we conclude that it is important to have both a relationship-based and capability-based access-control model in a smart home. Such a model should be flexible enough to address the complex needs and use cases that might occur.

B) **Child vs. Teenager**

Though both children and teenagers are under a parent or guardian's watch, a teenager (presented as 16 years old) and a child (presented as 8 years old) were given very different access scopes. After removing the five capabilities that are not applicable to a child (whom we assume lacks a smartphone), for twelve of the seventeen remaining capabilities teenagers were given greater access (FET, all $p < .05$). A 16-year-old teenager was regarded as a young adult by many participants and was more widely trusted to use capabilities responsibly. Therefore, the *always* permission was chosen often, and no need for supervision was mentioned in their free-text responses.

Meanwhile, granting an 8-year-old child unencumbered access worried participants much more. Some participants mentioned that they were concerned that a young child would misuse these capabilities, either intentionally or unintentionally, and thus ruin all the settings. Several participants even expressed their worries that a young child could get themselves in danger with the access. For instance, one participant, who selected *never* for the capability of seeing which door is currently locked or unlocked, wrote: *"An elementary school child should not be leaving the house on his own accord."* An 8-year-old child's level of understanding of a smart home system is also questionable. As a result, children rarely were granted access *always* for capabilities other than those related to lights.

Even for capabilities for which participants chose relatively restrictive settings for both teenagers and young children (e. g., "Order Online"), attitudes differed. Though only 5.3 % of participants agreed to give full access to "Order Online" to a teenager, 73.7 % chose *sometimes* over *never*, giving limited access to their teenager to buy things they needed on Amazon. For young children, 94.7 % participants believed that a child at that age should *never* have access to it, frequently justifying that there is no need for younger children to order things online themselves. Many participants mentioned supervision or limitations on what a teenager can buy on Amazon, but they did admit they would let a teenager buy things from Amazon themselves if they had a reason.

C) **Overall Preference for Restrictive Polices**

We found that, except for spouses and teenagers, most participants preferred a more restrictive access-control policy over a more permissive one. For nine of the twenty-two capabilities averaged over all relationships, more than half of participants chose *never* more frequently than *sometimes*, and *sometimes* more frequently than *always*. Averaged across all capabilities, only 18.1 % of participants ($\sigma = 0.12$, $median = 13.2\,\%$) chose *always* for visiting family members, 10.3 % for babysitters ($\sigma = 0.09$, $median = 7.9\,\%$), 8.3 % for children ($\sigma = 0.10$, $median = 5.6\,\%$) and 0.7 % for neighbors ($\sigma = 0.03$, $median = 0\,\%$). There was only a small group of capabilities for which participants were widely permissive: controlling lights and music, which do not have much potential to cause harm or damage.

Table 4.1: Potential default access-control policies that reflected the vast majority of participants' preferences.

All
- *Anyone* who is *currently at home* should always be *allowed* to adjust *lighting*
- *No one* should be *allowed* to *delete log files*

Spouse
- *Spouses* should *always* have access to *all capabilities*, except for deleting log files
- *No one except a spouse* should unconditionally be allowed to access administrative features
- *No one except a spouse* should unconditionally be allowed to make online purchases

Children in elementary school
- Elementary-school-age *children* should *never* be able to use capabilities *without supervision*

Visitors (babysitters, neighbors, and visiting family)
- *Visitors* should only be able to use any capabilities *while in the house*
- *Visitors* should *never* be allowed to use capabilities of *locks, doors, and cameras*
- *Babysitters* should only be able to *adjust the lighting and temperature*

4.4.4 Default Policies (RQ2)

In this section, we give an overview of the default deny/allow access policies we observed that capture most participants' responses. We categorize the policies by relationships and give an in-depth analysis of our findings.

Default Allow

A) **Spouses are Highly Trusted**

Averaged across all capabilities, 93.5 % of participants ($\sigma = 0.09$, $median = 95.3\,\%$) agreed to *always* give access to their spouse, while only 4.15 % ($\sigma = 0.05$, $median = 0\,\%$) answered *sometimes*, and 2.35 % ($\sigma = 0.06$, $median = 0\,\%$) said *never*. For participants who selected *always*, their most frequent reason was that they fully trust their spouse and that equality should be guaranteed in a marriage. Half of the non-permissive responses came from the capability to delete the smart lock's log file.

B) **Controlling Lights**

Access-control policies relating to lights were the most permissive. Looking at the responses for the capability to turn lights on and off, most responses align with a proposed default policy of people only being able to control the lights if they are physically present within the home. Relatedly, some participants chose *sometimes* for visiting family members and babysitters, depending on whether they are physically present within the home.

Default Deny

A) **Lock Log Sensitivity**

As mentioned in Section 4.4.2, "Delete Lock Log" is the capability least frequently permitted, and access should therefore be denied by default. Even for a spouse, this capability should not be accessed by default (only 68.4 % chose *always* for their spouse). More than 75 % of participants chose *never* for all other relationships. As the main method of retrospecting usage history, the log is not meant to be deleted.

B) **Supervising Children**

The elementary-school-age child (presented as 8 years old) was one of the most restricted relationships. On average across all capabilities, 69.4 % of participants chose *never* for the child ($\sigma = 0.19$, $median = 70.6\,\%$). Only neighbors received fewer permissions. In our chi-squared tests, we did not observe significant differences in desired access-control settings for children between participants who are currently living with a child, who have lived with a child before, and who have never lived with a child. None of our capabilities were considered child-friendly enough for even the majority of participants to *always* grant their elementary-school-age child access to that capability *always*. For only the "Light State" and "Play Music" capabilities was *never* chosen by fewer than half of participants. Despite being an immediate family member and living together, plenty of participants expressed fears that a child at that age might toy with these features and unintentionally mess up their settings or even cause danger to themselves. With supervision, though, many participants would consider giving temporary access to their children to gradually teach them how to use such a new technology.

C) **Ordering Online**

The capability to make an online purchase was generally limited to spouses only; 78.9 % of participants said that only their spouse should always be allowed to make online purchases, but 84.2 % also said that it was acceptable for non-spouse users to do the same if given explicit permission by the homeowner.

D) **Administrative Capabilities**

By default, only spouses should be able to access administrative capabilities, such as adding users, connecting new devices, and installing software updates. 89.7 % of participants gave their spouse access to these administrative capabilities *always*, while only 39.7 % of participants *always* gave comparable access to their teenage child. Unsurprisingly, under twenty percent of participants would give full access to other relationships.

4.4.5 The Impact of Context (RQ3)

Since there are many factors at play in the access-control-policy specification process, it is important to identify which contextual factors are most influential in this process and how they contribute to the final decision. The full results are visualized in Figure 4.3. We also ran chi-squared tests to see if each contextual factor had a relatively greater influence on some capabilities rather than others. While we did not observe significant differences for the "People Nearby", "Cost" and "Usage History" contextual factors across capabilities, we did observe significant differences for the other five contextual factors.

A) **Age**

The *age* of the user was the most influential factor on average across the twenty-one capabilities, except changing camera's angle (78.1 % on average, $\sigma = 0.13$, $median = 78.3\,\%$). The proportion of participants for whom age mattered varied across capabilities ($p = 0.040$). The main capability for which age played less of a role was for *changing the camera angle* (only

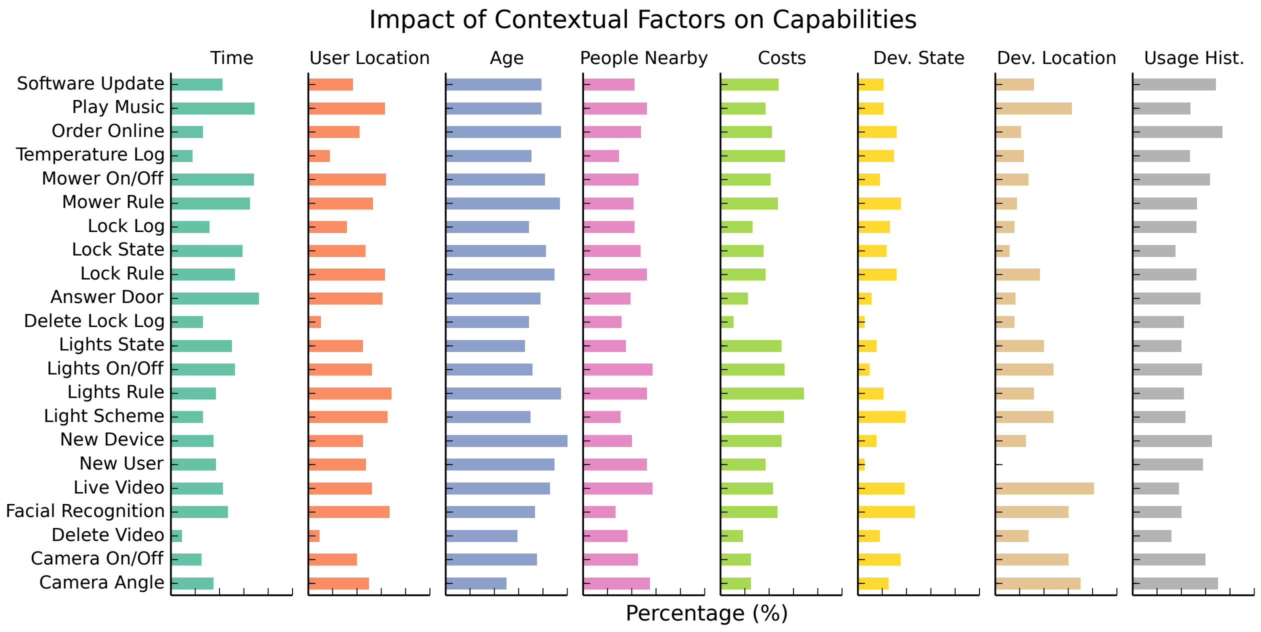


Figure 4.3: Contextual factors: Sometimes access must depend on the context. In the study we asked participants for such factors and identified multiple that are very influential (such as the age of the user) and learned how they contribute to the decision make process.

50 %). Many participants were concerned with letting a young person have access to certain capabilities. *"They need to be mature enough to use it responsibly"* was one typical response. However, another participant instead explained, *"It will be the person themselves and how capable they are with technology. I do not care about age."*. Thus, while *age* was frequently mentioned, in reality the decision process is more likely to be driven by how capable and responsible a user is, which sometimes correlates with the user's age. Our results indicate that a child at a young age (around 8 years old) is generally not perceived to be tech-savvy and responsible enough to be allowed unsupervised access.

B) **Location of Device**

The proportion of participants for whom the device's location impacted the access-control policy varied across capabilities ($p < 0.001$). Capabilities relating to cameras were unsurprisingly very location-sensitive. "Camera Angle" is the only capability for which a device's *location* was more frequently influential (70 % of participants) than the user's *age*. *Device* *location* was the second most frequently invoked factor for turning a camera on or off (60 %) and watching live video (81 %). If a smart camera is installed indoors, especially in a bedroom or bathroom, it will be much more privacy-sensitive. Participants reflected this by saying, for example, *"I can see where a guest/house-sitter/baby-sitter might need to access a view of outside or the garage but not inside."* Therefore, when designing a smart camera, whether the camera will be used indoors or outdoors should be considered and reflected in default access-control policies.

C) **Recent Usage History**

The proportion of participants for whom a device's recent usage history impacted their access-control policy did not differ significantly across capabilities. On average across capabilities, 51.7 % of participants ($\sigma = 0.12$, $median = 52.6\,\%$) agreed that this factor impacted their decision about the access-control policy. For participants who felt the device's recent usage history would change their decision, two main rationales arose. On the one hand, if the history states that a user is abusing a capability, then the owner may revoke access. One participant wrote, *"If someone were to misuse the device, you best bet they aren't getting a second chance. Alright maybe I'll give them a second chance, but definitely not a third!"*. On the other hand, if a user turns out to be trustworthy, then the owner may consider letting them keep the access, or even extending it. *"If my kid had been using the device responsibly, I would feel more comfortable giving them more access."* However, some participants felt the recent usage history was not particularly relevant for two main reasons. First, if the involved capability itself cannot cause much trouble, such as "Light Scheme," a common line of reasoning is that *"It would be hard to abuse this capability, so it doesn't matter to me."* Second, if the capability itself is so concerning that participants are reluctant to give others access (e.g., "Delete Video"), usage history did not play a role.

D) **Time of Day**

The importance of the *time of day* contextual factor varied across capabilities ($p = 0.001$).

"Play music" (68.4 %) and lawnmower-related capabilities (64.7 % for creating rules for the mower, 68.2 % for turning lawn mower on/off remotely) were particularly sensitive to the time of the day. In order to not interrupt other people's rest, participants tended to limit lawnmower usage usage to the daytime and playing music to the early evening.

E) **Location of User**

Capabilities that change devices' behaviors tended to be more sensitive to where the user is physically located when trying to control the device ($p < 0.001$) since many functionalities cannot be enjoyed without proximity. For example, creating rules that control the lights (68.4 % of participants felt the user's location mattered) and "Facial Recognition" (66.7 %) were prime examples. Many participants wrote that they would not want anyone who is not currently present in the house to use these capabilities unless it is the owner or their spouse.

F) **Costs**

The influence of the cost of exercising a capability did not vary across capabilities ($p = 0.162$). We believe this is in part due to our study design that did not include high-wattage appliances. Nevertheless, we observed some evidence of concerns with the cost of leaving lower-wattage devices, like lights, on during the day. Some participants mentioned that while lights do not consume a lot of electricity, cost can quickly become a concern if heavy appliances were to be involved. In addition, the influence of cost on online shopping differed due to different interpretations of cost. For cases where participants did indicate that cost is a concern, their interpretation was based on the cost of the good purchased, rather than the electricity used in placing an order.

G) **People Nearby**

43.6% of participants ($\sigma = 0.09$, $median = 43.6\,\%$) indicated that who else is nearby might impact their access-control decision. The role of people nearby did not differ significantly across capabilities ($p = 0.400$). For participants who believe this factor matters, there are two contrasting conclusions. Some people might feel more permissive when they themselves are around since that means they can supervise everything. However, others felt less permissive because if they are around, there is no need for others to have access since the others simply would need to ask the owner. Therefore, it is important for the system configuration to take these divergent mental models into consideration, letting users decide which direction they might choose to go in.

H) **State of Device**

The current state of device was overall the least important factor in participants' access-control decisions on average ($mean = 23.7\,\%$, $\sigma = 0.11$, $median = 22.3\,\%$), though this importance did differ across capabilities ($p = 0.044$). Notably, 46.7 % of participants who answered about the "Facial Recognition" the capability marked the state of the device as an influential factor. This is because if the camera is currently off, then there is no reason for anyone to enable of disable the facial recognition.

I) **Other Factors**

We included a free-text question with which participants could list other factors they thought played a role in their access-control-policy specification process. In their responses, we observed a long tail of additional contextual factors, including weather, people's familiarity with technology, how close they are to the owners, and the frequency of one's access to a certain capability.

4.4.6 Wrong Decisions' Consequences (RQ4)

Analyzing consequences of incorrect authorization decisions, we can learn how much tolerance a user has for a policy to fail given a specific capability and relationship pair. It is crucial to understand how strongly users would feel if the system were to malfunction. We analyze *false allow* and *false deny* decisions separately.

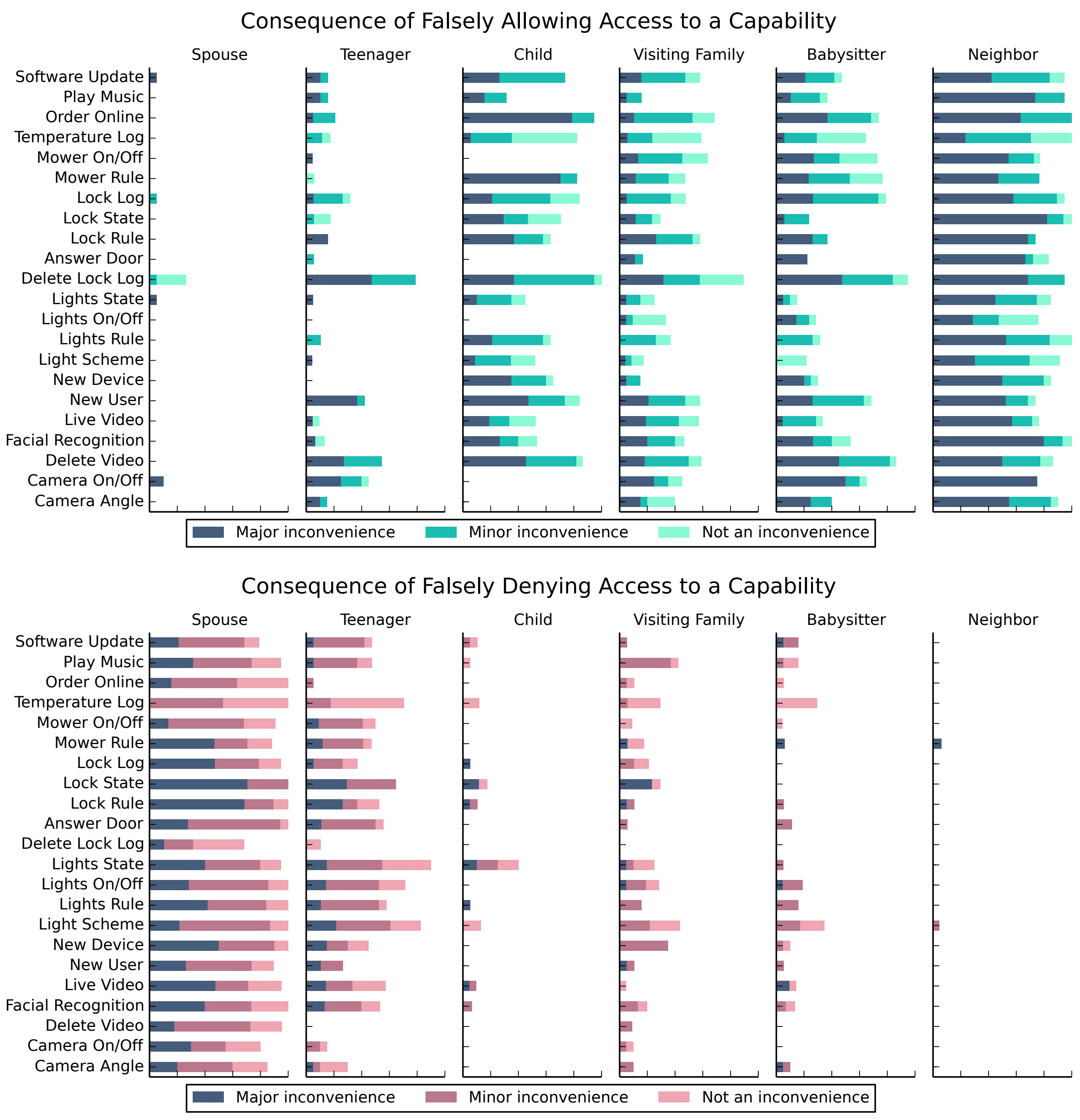


Figure 4.4: Perceived consequences of incorrectly allowing someone to use a capability when they should never be permitted to do so (top) or incorrectly denying someone when they should always be permitted to do so (bottom).

False Allow

Note that responses about *falsely allowing access* belong to those participants who intended never to grant access to a certain capability to a certain relationship. These participants therefore might be more concerned than other participants in certain aspects, which leads to some narrow tensions with the broader trends seen in previous sections. Figure 4.4 (top) summarizes these results.

A) **Neighbor false allows a major inconvenience**

Across all capabilities, 64.1 % of the participants stated that it is a *major inconvenience* if the authorization system gives access to their neighbor by accident. Turning the security camera on or off (100 % a major inconvenience) and creating rules for a smart lock (92.9 % a major inconvenience and 7.1 % a minor inconvenience) are the most concerning capabilities. Note that in the study, we described the people representing the relationship *neighbor* as "good people, which includes friendly small talk, and occasional dinner invitations." Nevertheless, privacy and security were major concerns.

B) **Spousal false allows have severe consequences**

Though the number of false-allow responses for the spouse relationship is quite small ($n = 10$), it still gives some interesting insights. 50 % of the answers are based on deleting log files from a smart lock. Four out of five respondents rate falsely allowing a spouse to delete the log file not to be an inconvenience. *"I wouldn't really care about my spouse deleting it, but it would bother me that the system is not secure,"* was a typical response.

There were five more responses from other capabilities. From those, four out of five indicated that a false allow decision was a major inconvenience. It is surprising to see that a few participants believed it a major issue if the mechanism allows their spouse to access certain capabilities by mistake.

C) **Visiting family false allows a minor issue**

Though we presented earlier that participants' permissiveness toward a visiting family member and a babysitter was very similar (and tended toward not being permissive), we observed a distinction when it comes to false allows. Participants were much less concerned with incorrectly giving access to a visiting family member (70 % chose *minor* or *not an inconvenience*) than to a babysitter (58 %). Responses like *“He is my family member so I trust him a bit”* were common. While participants believed the visiting family member would not do much harm, false allows would still upset them a bit.

D) **Shopping / lawn mowers forbidden for children**

Among all capabilities, incorrectly allowing a young child to order online (79 % a *major inconvenience*) and create rules for the lawn mower (70.6 %) were the two capabilities where false allows for a child raised great concern. A child at such a young age is generally not trusted with ordering things online. *“The child could spend a ton of money on products we don’t need,”* wrote one participant. A lawn mower is considered dangerous. One participant simply wrote, *“(A lawn mower) could cause harm to the child.”*.

False Deny

Responses in this section, *falsely denying access*, come from participants who intended to give access to a certain relationship. Figure 4.4 (bottom) visualizes the full results.

A) **Participants Did Not Want to be Locked Out**

Lock-related capabilities raised the most concern (63.9 % of responses for “Lock State” and 58.8 % for “Lock Rule” found falsely denying access *major inconveniences*). Participants tended to be very cautious about smart locks. Even though viewing a lock state does not directly concern locking or unlocking the door, participants still worried whether a malfunctioning access-control system would lock people out, thus marking these false denies as major inconveniences.

B) **Spouses and Trust Issues**

One common reason why participants gave full access to their spouse is because they believe two people in a marriage should be equal, which means two parties should have the same access to a system. Therefore, if their spouse is accidentally rejected by the system, it could raise trust issues and spur arguments within the marriage. We found a number of responses similar to *"I would not want my spouse to think I did not trust them."* It is interesting to see that not only do relationships impact access-control policies, but relationships are also influenced by authorization results. Thus, extra care is required for such relationships.

4.5 Conclusion

Capabilities, Relationships, and Context. While access control in smart homes is currently often device-centric, our user study demonstrated that a capability- and relationship-centric model more closely fits user expectations. Home IoT technologies allow for multiple ways of achieving the same end result, whereas devices often bring together vastly different capabilities. For example, to increase a room's brightness, one could remotely turn on a light using a smartphone app, remotely open the shades, or ask a voice assistant to do either. This model reveals nuances that are missed in the device-centric model. From the data for RQ1, we see that the desired policies can vary widely within a *single* device based on the relationship and the context of access. Although some of these distinctions are intuitive (e. g., child vs. teenager), others are more nuanced and surprising (e. g., babysitter vs. visiting family member). They also provide a concrete access-control vocabulary for developers of future smart-home devices.

A difficult decision in access-control systems involves default policies. In multi-user social environments, intuition suggests a default policy would be complex. Surprisingly, our data for RQ2 suggests that potential default policies are actually simple and reminiscent of non-IoT policies. For example, our default policy says that a person can actuate a light if they are physically close to it. Though IoT lights can be remotely actuated, the relation between proximity and using a light is not broken. Although conceptually simple, this rule's enforcement is non-trivial, requiring creating and deploying authentication methods beyond the possession of a smartphone.

Data from RQ3 suggests that the factors affecting access-control decisions are heavily context-dependent. Current home IoT devices only support rudimentary forms of context (Section 4.1). Some contextual factors, such as age, are currently present in smartphones and cloud services (e. g., *Apple's iCloud Family Sharing* supports adding a child Apple ID that requires parental approval for purchases, while Netflix has *kids* option). We recommend that for home IoT settings, these contextual factors should be a first-order primitive.

Based on these findings (RQ1-3), we envision several changes to smart-home setup. This process currently involves installing hubs and devices with a set of coarse-grained accounts. Our work suggests that future smart homes could instead set access-control policies by walking users through a questionnaire whose vocabulary derives from our user study. This is closer to the experience of setting up software, where a package comes with secure defaults that are customized to the specific installation. Using default policies derived from our results would minimize user burden since it would reflect common opinions by default. Physical control (e.g., switches) already enables certain default policies, so software authorization might seem unnecessary in certain situations. However, switches are often add-ons to IoT starter kits, making software authorization a prerequisite to a satisfying user experience.

Authorization Vocabulary. Based on our study results, we discuss a potential authorization vocabulary that is helpful in building future authorization and authentication for home IoT platforms. The basic unit of the vocabulary is a triplet containing <Capability, UserType, Context>. As discussed, capabilities better capture the nuances of access control in the home than devices. UserType captures the relationship of the user to the home, and to the owners. From our study, these types should nonexhaustively include: Spouse, Teenager, Child, Babysitter, and Neighbor. Spouses tend to be users with the highest levels of access, generally equivalent to administrators in traditional computing systems. Context refers to the environmental factors that might affect an access-control decision. For example, certain parents might be more permissive in allowing a child to watch TV without supervision. Based on our study, at the minimum context should include: Time, User Location, Age, People

Nearby, Cost of Resource, Device State, Device Location, and Usage History. Depending on the Capability and the UserType components of the triplet, the importance of the context can change. For example, for a UserType of Child, the 'People Nearby' contextual factor plays a prominent role in the access-control decision. However, for spouses, it generally has no bearing. The same goes for the Capability. The 'Device Location' contextual factor is crucial for camera-related capabilities, but not so important for the capability of adding a new user.

Mapping Authorization and Authentication. Although we focused on analyzing access control, we briefly discuss how our findings affect the design of authentication mechanisms. Below, we discuss a set of authentication mechanisms and comment on their ability to identify users, relationships, and contextual factors. We also discuss privacy limitations and the effect of false positive and negatives.

Smartphones are the most widely used devices to access IoT devices in the home. Users may present their identity to a device using a password, PIN, or (more recently) fingerprints. These identities can be used by home IoT devices to determine the identity, and hence relationship, of the person attempting access. From the perspective of false positives/negatives, smartphones can closely match user expectations. They are inconvenient, however, for temporary visitors because they require the visitor to install an app and the owner to authorize them.

Wearable devices like watches, glasses, and even clothing [115] might serve as proxy devices with more natural interactions than a smartphone. For example, a user can gesture at a nearby device to control it (e. g., wave at a light to turn it on or off). As each user will perform a gesture differently, it can also serve as a form of authentication and thus be used to identify a person and their relationship. Furthermore, the proximity of a wearable device is helpful in identifying several contextual factors, including user location and nearby people. From a false positive/negative perspective, biometrics require quite a bit of tuning that can affect an owner's choice of using this method, especially when authenticating high-access spouses or for operating dangerous equipment like lawn mowers.

Voice assistants are increasingly ubiquitous in homes. Although such assistants can perform speaker identification (e.g., Google Home Voice Match), they are currently used as a personalization hint rather than a security boundary. However, future versions that use additional hardware might be useful in determining a speaker's identity and relationship for access-control purposes [48]. Such assistants could help identify contextual factors like the location of a user or the presence of nearby people (e.g., a supervising adult near children). From the perspective of false positives/negatives, any voice-based method will require tuning. Audio is especially sensitive to background noise. Audio authentication also introduces privacy issues, as well as the potential for eavesdropping and replay attacks. Advances in computer vision can also be leveraged to identify users, their relationship, and their location within a home with cameras. However, it is possible for computer vision systems to falsely identify individuals or confuse identities. Thus, some level of false positive/negative tuning will be required, especially when a household is expected to have many temporary occupants. A big downside of this mechanism is the privacy risk—cameras can track home activity at a high level of granularity. However, some of the privacy issues could potentially be alleviated using local processing or privacy-preserving vision algorithms [116].

Bilateral or continuous authentication mechanisms embody the idea that a user has to be: (a) physically present, and (b) currently using the device [117, 118]. Such mechanisms are readily able to identify users and relationships, and to support contextual factors involving user presence. False positive/negative tuning varies based on the specific instantiation. If a wearable device with a continuous authentication algorithm is used, then the false positive/negative rates must be considered. Privacy concerns can be alleviated if this mechanism is implemented in a decentralized manner—only the user's proxy device and the target device are involved in establishing an authenticated channel. It can also provide a simple solution to the de-authentication problem (revoking access if a temporary visitor is no longer welcome).

In sum, we have taken initial steps toward reenvisioning access-control specification and authentication in the home IoT. Much work remains in continuing to translate these observations to fully usable prototypes, as well as in supporting ever richer capabilities and interactions.

CHAPTER 5
CONTEXT SENSING FOR ACCESS CONTROL IN THE ADVERSARIAL HOME IOT

As discussed in the previous chapter, in home IoT, the set of users who ought to have access to a resource varies over time and may include guests, in-home workers, and others [2, 41, 42]. Desired access control in smart homes is frequently *contextual* (situational). Rather than granting unconditional access to a given user or a given role, authorization decisions may depend on the context. A context can be the user's location relative to the device, the history of the user's interactions with the device, or the state of the home [8]. An example policy is that a child can only use the smart TV when a parent is nearby [8]. Here, the system must verify two contexts: (i) a child is trying to use the TV and (ii) a parent is around. Enforcing contextual access control requires privacy-preserving and trustworthy context sensing. That is, a *sensor* (e.g., a motion sensor) must reliably detect some *context* (e.g., a room is unoccupied) while respecting users' privacy.

Prior work in the security and privacy community has already proposed ways to utilize contexts in access control [43, 119], but has not focused on how to detect contexts in the physical world in ways that are both trustworthy and privacy-preserving. A large amount of existing work on sensing and ubiquitous computing could be applied here, but it mostly ignores attacks, adversaries, and privacy. For example, work done on robust sensing often sacrifices privacy by adopting more invasive sensing methods [120] or denser sensor deployment [44, 121]. This is not realistic for an intimate setting like one's home. Some bodies of work also discover that errors are bound to occur in particular circumstances, but they regard these errors as rare or unintentional occurrences [122, 123, 124]. Adversaries can exploit this assumption.

In this chapter, we critically reevaluate the literature on context sensing in homes with a security and privacy mindset. Furthermore, we translate this literature to the problem of context sensing for access control, identifying sensor types that best match specific contexts within practical constraints. To do so, we first identified home contexts that are critical to access control from the small literature on contextual access control in smart homes. We then systematically searched the proceedings of the last decade of top conferences in sensing systems (SenSys, MobiSys, and MobiCom), ubiquitous computing (UbiComp/IMWUT), and human-computer interaction (CHI and UIST), identifying dozens of recent papers about sensors that can detect those contexts in smart homes. To capture well-known mature sensors, we also searched for commercially available sensors for smart homes and added classic papers on relevant sensors. This process left us with 94 pairs of contexts and sensors. Analyzing these papers while also revisiting key IoT papers from the security, HCI, and usable security literatures, we constructed a decision framework that highlights each sensor's pros and cons for security, privacy, and usability when used to detect an access-control-relevant context in a smart home. Our work thus lays a foundation for secure, practical, and privacy-preserving context sensing in smart homes.

We first create a novel threat model broadening the adversaries that prior literature has considered for smart home sensing. Prior work has focused on how experts can exploit IoT systems through software vulnerabilities [7, 5], default passwords [31], replication of physical traits [125], and adversarial examples [95, 18, 20, 19]. While our model encompasses these threats, we focus on non-technical adversaries with legitimate access to a home, such as kids, roommates, guests, and workers, who usually have stronger motivations than remote strangers. Notably, most papers on context awareness and home sensing do not consider the adversarial mindset typical in the security community.

From our threat model, we make several observations. First, physical denial-of-service attacks are trivial against many sensors. Thus, in contextual access control, policies that allow access by default or rely on the absence (rather than presence) of a characteristic are easy to bypass. Second, non-technical users are highly capable of replay, imitation, and shoulder-surfing attacks. They can also impersonate someone by simply taking that person's phone. Identity cannot be reliably authenticated through possession of a phone or naive recognition of voices/faces.

Contextual access control in homes thus requires deploying sensors with key properties. The sensors, alone or in ensemble [44], must resist attacks from both technically literate outsiders and non-technical insiders. They must also minimize inadvertent data collection because sensors may be deployed in private areas of the home. Finally, household members must find the sensors acceptable.

Then, we develop a decision framework for evaluating the degree to which a particular sensor possesses these key security, privacy, and usability properties. We further distinguish between attacks of different complexities, privacy considerations from various actors, and specific usability criteria. The latter includes ease of deployment, reusability of a sensor across contexts, and inclusiveness. This framework will be useful for individuals who design or deploy sensors in homes, including DIY users [126], manufacturers, and researchers in security and in sensing. We will refer to these individuals as *smart home designers.* This framework can help smart home designers navigate the vast array of sensing mechanisms described in the literature or available commercially. We envision the framework helping smart home owners to decide which sensor to use, manufacturers to design their products for facilitating contextual access control, and researchers to develop sensors that are more sensitive to security and privacy issues. The framework also outlines criteria to consider when designing a new sensor. In particular, our framework elucidates key trade-offs among the variety of sensors (e.g., motion sensors, microphones, thermal imaging) that can detect a given context (e.g., whether anyone is in a room).

Eventually, we apply our framework to highlight trade-offs in deploying sensors for access control in homes. Through a systematic review of the sensing literature, we identify *indicators* (e.g., characteristics, such as gait) and associated *sensors* (e.g., a pressure sensor mat for detecting gait) for sensing either *identity* (e.g., this is Jane) or *context* (e.g., this is an adult). Using our decision framework, we evaluate each sensor's key properties. We used our literature review to gauge sensors' robustness to attack, privacy properties (e.g., requirements for data storage), and deployability. With our framework, smart home designers can identify the sensors that support desired contexts for access control and recognize trade-offs in security, privacy, and usability. **To keep our framework and evaluations up-to-date, we have released them in a public GitHub repository.**[1] Researchers may publicly modify, expand, or dispute the table through pull requests and issues, facilitating open discussion between the sensing and security communities.

Applying our framework yields the following insights. First, we find that many current sensors, when used alone, do not adequately address potential threats from non-technical adversaries. They are especially vulnerable against rarely studied physical DoS attacks. Second, many sensors collect more data than needed. Contrary to currently deployed architectures, many sensors do not require cloud storage for data. Lastly, we found that many sensors are not inclusive based on age or disability, and some can be ineffective under certain environmental factors.

5.1 Our Threat Model

Sensor-based access control in homes requires robust sensing that protects user privacy. Prior IoT research has primarily focused on defending against remote attacks against IoT software [7, 6]. However, local attackers—regardless of technical background—can also pose a significant threat to the system by tricking physical sensors into detecting incorrect contexts or violating others' privacy. In fact, potential local attackers like family members, roommates, guests, and workers could have stronger motivations to bypass access control than unacquainted remote attackers. Our work examines local threats broadly and focuses

1. `https://github.com/UChicagoSUPERgroup/eurosp21`

on those posed by non-technical users with legitimate or illegitimate access to a home. Below, we taxonomize goals, attacks, and attackers. In light of the larger literature on context sensing, we revisit these attacks within our decision framework (Section 5.2).

5.1.1 The Attacker's Goals

One of our key insights is that non-technical attackers with modest and localized goals are a threat to contextual access control. Whereas remote attackers disrupt at scale, non-technical local attackers might only want to gain illegitimate access to some resource or spy on another individual. For example, a child may wish to watch TV without approval, a burglar may want to erase security camera footage after committing theft, or (as can be the case with intimate partner violence [127, 111]) an abusive member of the household may try to spy on members of their household by evading policies stopping security cameras from recording when people are home.

Local attackers might aim to bypass access control or compromise the privacy of others in the home.

Strategies for attacking sensors depend on the policy. A *default-deny* policy, which automatically denies access to unknown users, is not always advisable. For instance, prior work found users prefer default-deny policies for locks, but would rather permit unauthorized users to control smart lights than leave users in the dark [8].

Impersonation: Under a default-deny policy, a system only accepts authorized and authenticated users. An attacker must impersonate an authorized user or fabricate a valid token through imitation or replay attacks.

We find that these attacks often do not require technical knowledge (Section 5.4), especially in an intimate setting like a home where boundaries to privacy are reduced and private resources are easy to acquire. For example, many widely deployed facial-recognition systems lack depth or liveness detection. One can trick them by presenting a photo or video of an authorized user [128]. Photos of authorized users (e.g., a child's parents) are easy to find in a home, and videos can be taken in secret.

Similar issues arise for audio. People with access to a home can record authorized individuals speaking to voice interfaces. While authenticated speaker recognition is an active area of research [129], many widely deployed voice interfaces are vulnerable to simple replay attacks [99, 100] or even lack authentication entirely [17].[2] Off-the-shelf voice morphing compounds this problem [130].

Local attackers have extensive access to photos and audio, making basic face or speaker recognition systems vulnerable to replay and imitation attacks.

Current home IoT systems tend to rely on smartphones as a proxy for identity, capitalizing on their ubiquity. However, smartphones often run out of battery, and they do not offer the convenience of other interaction modalities (Chapter 3). This practice also falsely assumes that the user is always near their phone. For example, if the smart TV will turn on only if an adult's phone is in the room, a mischievous child can take their parent's phone while the parent is sleeping. Furthermore, smartphone authentication is still not fool-proof as it is often knowledge-based (e.g., PINs). It is often easy for others in the home to bypass this authentication through shoulder-surfing.

Existing practices of using phones (potentially with authentication) as a proxy for identity in shared spaces can be risky in terms of both security and usability.

Invisibility: Contextual access-control policies can also allow access by default. One example would be using the smart stove. Whereas visitors or babysitters may be allowed to use the stove, a child should not use it for safety reasons. A natural policy that follows is "anyone except a child can turn on the stove." When these *default-allow* policies depend on

2. In our informal testing, Google Home's speaker recognition only seemed to verify the person who said "OK, Google." It accepted further commands spoken by someone else, making replay attacks trivial.

Dimension	Type	Capabilities	Examples
Access	Indoors	Physical access to indoor & outdoor devices/sensors Rich observation opportunities Full knowledge of sensor models & locations Knowledge of access-control policies & automations	Family member, babysitter
	Outdoors	Physical access only to outdoor devices/sensors Limited observation opportunities Opportunistic attacks that reach more victims	Neighbor, prospective burglar
Expertise	Expert	Sophisticated network and imitation attacks Ability to craft black-box adversarial examples Unsophisticated replay/imitation attacks, block sensor	IT professional, hacker
	Non-expert	Unsophisticated replay/imitation attacks, block sensor	Child, domestic worker
Resemblance	Similar	Spoofing (through imitation) Higher possibility of inadvertent false positives	Sibling, one who looks similar

Table 5.1: Local attackers can be characterized along the dimensions above, impacting attack capabilities.

not sensing a characteristic or situation, e.g. "record security video of the bedroom when *no one is home*), an attacker needs nothing more than to make the characteristic or situation "invisible." They can do this by changing or blocking the sensor's field of view.

We will refer to such attacks, where the local attacker prevents the sensor from physically detecting a context, as *physical denial of service (DoS).* This can entail blocking a motion sensor with paper or overloading a microphone with loud noise (including outside the human hearing range [131, 19]). Sensors must detect whether they are receiving accurate and fresh input.

Default-allow policies, which rely on *not* detecting a given situation, can be defeated by blocking sensors.

5.1.2 Attacks

Based on these attacker goals, we surveyed top security and sensing conferences to identify likely attacks. We clustered prior work based on attack method, resulting in three major types of attacks: 1) replay and spoofing attacks; 2) adversarial examples; 3) sensor hardware attacks. Note that replay and spoofing attacks differ in practicality despite often appearing together in the literature. We did not find mentions of physical DoS attacks in our literature survey, but include them because they are a clear threat to access control. Below, we define these attacks.

Replay Attack: The attacker collects a credential and feeds it back to a sensor. For example, the attacker can play a voice recording, show a photo of a face, or make a gummy mold of a specific fingerprint [125]. Our focus in this SoK is on replaying the physical signal itself, although network traffic can sometimes also be replayed.

Spoofing: The attacker forges an approximate credential or situation they have not necessarily captured. Smoke can spoof a fire, and energetic pet cats can spoof occupancy.

Physical Denial of Service (DoS): Jamming, blocking, or moving a sensor can prevent accurate sensing. It is important to note that the sensor detecting the *absence* of a characteristic or situation is different from *not* detecting it. For instance, when trying to sense whether a room is empty, a camera blocked by a piece of paper will not detect any people. This differs from a camera affirmatively seeing a room without people. These attacks are often easy to deploy, but have not yet received much attention.

Adversarial Examples: Against ML-based sensing methods, the attacker can poison the training data or add carefully crafted noise to inputs [95, 132].

Sensor Hardware Attacks: The attacker leverages the physical principle behind the hardware to deceive the sensor, such as with signal injection attacks [19, 133].

Inadvertent False Positives: This is not quite an attack, but a sensor incorrectly detecting an identity or situation can still compromise access control.

5.1.3 Physical Sensors' Potential Attackers

To understand each attack's feasibility, we characterize the attacker's capabilities. Table 5.1 provides a summary. Our threat model concerns attackers who *violate* access-control policies. We thus ignore adversaries who *create* unreasonable policies, such as domestic abusers attempting to spy on their family. Defending against those adversaries requires countermeasures beyond access control.

Access: An attacker with access to the home would be well-positioned for physical attacks against sensors. They can observe authentication processes in the home, potentially repeatedly, to record information for replay or imitation attacks. For example, a roommate might encounter multiple instances of the user speaking to a voice assistant. They thus have multiple opportunities to record the user's voice for tricking speaker-recognition algorithms. By having access to the home, attackers can also infer access-control policies, automations, and sensor locations or types from their observations. Legitimate access can be permanent, such as for residents, or temporary, such as for visitors and domestic workers. Illegitimate access occurs when people enter the home without permission.

It is also possible for attackers to access sensors outside the home [134, 13] or make inferences using partial information (e.g., from sensors visible through windows). Some individuals who might rely on these methods include neighbors and prospective burglars. We note that modeling the attack surface cannot rely on a simple indoor versus outdoor dichotomy. For example, one can control a voice assistant through an open window.

Expertise: Attackers with technical expertise, such as infosec professionals, are capable of sophisticated attacks. Some attacks against ML-based sensor systems are of this nature. They can involve carefully crafted eyeglasses [95], stickers [18], or audio [131, 19]. Experts can also target sensors' physical principles, such as applying acoustic interference to accelerometers [135]. Finally, network- and software-based attacks are also possible.

On the other hand, nontechnical attackers can carry out replay or imitation attacks that only require observations (e.g., spoken passwords) or commodity recording equipment (e.g., a smartphone). They can also disable sensors by blocking, repositioning, or unplugging them.

Resemblance: Biometric sensors may confuse individuals of similar physical traits. Biological family members often share physical resemblances and have easy access to sensors because they often live together or visit each other. Real-world examples include one man who tricked a voice-recognition system by imitating his twin's voice [99]. Identical twins can also fool facial recognition [136]. It may also be possible for unrelated people with physical resemblances to trick the sensors.

Our threat model highlights two key ideas missing from prior work. First, most work focuses on threats from attackers with extensive resources and expertise. We show that non-experts with access to the home are capable of replay and spoofing attacks against sensors that support contextual access control. Second, blocking sensors can allow attackers to evade some access-control policies. This method of attack has not yet been studied extensively.

Contextual access control must consider that non-experts with access to a home can attack sensors.

5.2 Decision Framework for Context Sensing

Individuals designing or deploying home sensors need a framework that helps them navigate the trade-offs between sensors' security, privacy, and usability properties in conjunction with the users' needs and the space itself [45]. These individuals, whom we term *smart home designers*, will benefit from the framework in different ways:

- Do-it-yourself smart home owners can learn security and privacy implications of selecting certain sensors.
- Sensor manufacturers can holistically evaluate their current sensors' trade-offs and identify additional contexts that need new sensors to be developed.
- Security and sensing researchers can identify security and privacy gaps that guide their future research.

For example, a smart home owner might wish to know when anyone is at home. Consulting our framework reveals that cameras are suitable for this, but are not privacy-preserving.

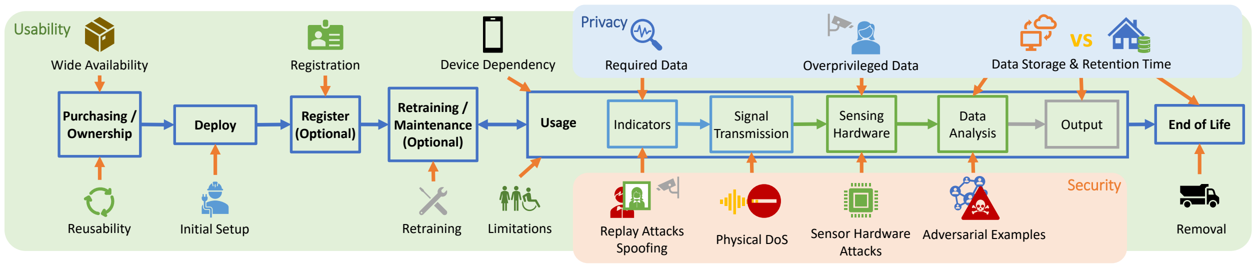


Figure 5.1: Different issues emerge in difference stages of using sensors in home.

Meanwhile, pressure sensors on the floor would be privacy-preserving, but are impractical and expensive to install. The user can now determine whether to prioritize occupancy detection at the cost of privacy.

Here, we first explore the life cycles of adopting a sensing technique. Then, for each stage of the life cycle, we further define the main security, privacy, and usability criteria that smart home designers must consider in choosing sensors, which we collectively consider our framework. We constructed this framework by critically analyzing the 94 pairs of sensors and contexts we identified through our systematic review of the sensing literature (see Section 5.3.2) relative to the security and usable security literatures concerning the home IoT. We also considered broader security principles to fill in potential gaps in this framework.[3]

5.2.1 Life cycles

Adopting a new sensing technology in one's home is a long-term and ongoing process. To avoid missing crucial challenges during the process, we first define different stages of the adoption process, as depicted in Figure 5.1.

Acquiring the required hardware: A user might need to buy new sensors, which is a financial and time investment.

Deploying the hardware: After acquiring the hardware, users need to install it in their

3. The team that constructed the framework included multiple students and three faculty members. Two of the faculty members focus on security and privacy research, but also have experience with machine learning research. The other faculty member conducts sensing research.

homes. When needed, users might also re-deploy hardware, such as to reposition it.

Registration (optional): Sometimes the hardware may require the user to register themselves first, which is especially common for sensors pertaining to an identity.

(Re)training / Maintenance (optional): Before usage, machine learning-based sensing methods commonly require the user to train the model about the context in its unique environment. Retraining may also be required in the future to adapt to users and a sensor's environment changing over time. Maintenance, such as battery replacement and routine check-ups, may also be required.

Usage: After training, the sensor is ready for use. We expect the sensing technique to operate until the user stops using the sensor. To identify possible issues in this stage, we must abstract how the sensing technique works.

Sensing detects environmental events, such as temperature changes, movement in the background, and sound. We term these *indicators*, which could be mapped to a context. For example, if a sensor detects movement of a heat source, it is likely to be someone moving nearby.

To detect the indicator, the sensor needs a signal sent or radiated from the source. Depending on how far the signal can be transmitted, the sensor may require direct contact, near-field communication, or far-field communication. We term this process *signal transmission.*

Once the sensing hardware receives the signal, it first needs to process the analog signal, such as using amplification and noise filtering. The analog signal can then be converted into a digital signal for further processing. The *sensing hardware* stage represents the above process.

Finally, the digital signal, or the raw sensor data, is sent to a processor or the cloud for further computation. Depending on what the sensing method is designed for, different *data analysis* methods may apply here. For example, facial recognition and gait recognition may both rely on cameras, but the data analysis would differ. Once the sensed data analysis is complete, the algorithm *outputs* whether the context it aims to detect is active.

End of Life: The user may eventually decide to uninstall the home sensor. In this stage, the sensor may be directly thrown away, given to others, or sent back to the manufacturer for upgrading or replacing. The hardware is not guaranteed to be properly destroyed. Thus, information leakage after disposal is possible. We treat the uninstallation process as two parts: removing all data (e.g., factory reset) and physically removing the sensor from the home.

5.2.2 Security

We consider two ways in which a sensor may be attacked. One is through *inadvertent failures.* An attacker may bypass an error-prone sensor through brute force. The other is through *intentional attacks.* These attacks are described in detail in Section 5.1. Figure 5.1 also indicates at which stage these attacks might occur.

We do not consider attacks before the usage stage. The set-up stage occurs only once and the victim is often present, increasing the difficulty of attacking the sensor itself. Therefore, during the set-up stage, it is more likely for the attacker to perform network attacks (e.g., sniffing, person-in-the-middle), which are out of this thesis's scope.

In Tables 5.2-5.4, a red "!" signifies that a sensor is easily susceptible to a given attack. A yellow "?" signifies that it is not very susceptible to the attack. If no symbol is shown in the table, the attack is implausible against the sensor (e.g., replay attacks against smoke detectors).

5.2.3 Privacy

Sensors collect data to operate, but excessive collection of sensitive data causes privacy concerns. Furthermore, certain contexts require intensive computation on data that is collected over long periods of time. To identify potential privacy threats during the usage stage, we review each stage carefully to identify general threats. We assume that the sensing software is secure and do not consider privacy threats before the usage stage. Our framework considers the following aspects:

Required Data: Data that must be collected for the sensor to function. Depending on which *indicator* the sensor detects, different types of data are collected, with various privacy implications.

Overprivileged Data: Depending on which sensor the designer decides to use, superfluous data might be collected inadvertently. For example, a microphone for occupancy detection also records conversations. In the "overprivileged data" column of Tables 5.2-5.4, *poor* means the sensor collects unnecessary and sensitive information, *acceptable* means it collects unnecessary data that is not sensitive, and *good* means it does not collect superfluous data.

Data Storage: Data must be analyzed and stored in the cloud if the device lacks the computational power or storage space for local processing. For other sensors, however, data can be stored on the device containing the sensor or on an in-home hub. Nonetheless, companies often upload data to the cloud even when unnecessary [137]. There is no guarantee that the uploaded data will be used ethically [138], which can deter users from deploying some sensors in homes [45]. We consider whether each sensor's data *must* be stored on the *cloud*, or whether *local* storage supports the needed functionality. We leave out of scope the question of whether a company will choose to upload data to the cloud even when it could be retained locally.

Retention Time: Some sensors require longitudinal data (e.g., for training a model). Companies may again decide to store all data indefinitely even when not strictly necessary. *Transient* storage means sensed data can be immediately discarded, while *persistent* means it must be retained until the user factory resets the device. Similar to *data storage*, companies may retain users' data for as long as they want, even if the user factory resets their device and deletes their account. To focus on the requirement for enabling the sensing technology, we only consider how long the data must be available for the functionality.

5.2.4 Usability

To assess a sensor's usability for a non-technical end user, we consider the following criteria, which we compiled based on the stages identified in Figure 5.1.

Wide Availability: Users are more likely to adopt sensors that they can easily acquire. For example, one can sense occupancy with motion sensors or ultrasonic sensors, but users and designers may prefer the former because of their cheap cost and ubiquity. Nonetheless, more expensive sensors (e.g., cameras) may also be widely available if they fulfill multiple use cases. This may benefit users because sensors that fulfill multiple use cases may obviate the purchase of additional sensors.

Initial Set-up: How difficult is it for a non-technical user to set up the hardware during the deployment stage? *Good* means little to no effort is required, such as plug-and-play installation. *Poor* requires substantial effort from the user, such as renovating their current home for installation (e.g., painting the wall, changing the floor). Anything between *good* and *poor* was deemed *acceptable*.

Registration: How much effort does it take to register a user, or how long does it take to collect enough data to train the model? *Good* means no registration or training is needed. *Acceptable* encompasses two situations. In the first situation, the sensing method requires straightforward registration or data collection, meaning registration should not take over 10 minutes. This includes most commercial products, such as Touch ID or Face ID. In the second situation, data collection needs more time to finish, but does not require user attention. For example, a system from Hsu et al. [123] required the user to wear an accelerometer for days as ground truth for identifying the user from their RF reflection. While this process takes days, no attention is required, earning it an *acceptable* rating. *Poor* takes significant effort from users, usually exceeding 10 minutes in duration while requiring constant attention the entire time. For example, Qian et al.'s system [139] requires the user to walk for four minutes each at three different paces.

Retraining / Maintenance: How often is model retraining or hardware maintenance required? *Good* requires none. *Acceptable* requires occasional retraining or maintenance less than once a month (e.g., changing batteries every few months). *Poor* requires retraining or maintenance at least once a month. When evaluating biometric sensors, we assume an adult user with stable features.

Reusability: Some sensors can detect multiple contexts. For example, cameras can detect age, room occupancy, or an identity. *Good* means many contexts can be sensed, as with cameras. *Acceptable* means a few contexts can be sensed, as with radar sensors. *Poor* means the sensor detects only one context, as with fingerprint sensors.

Device Dependency: Some methods require users to carry a device (e.g., a phone) during usage. *Good* means no such device is required. *Poor* means that it is required.

Limitations: We consider whether the sensor is effective for all groups of users and under all situations. We focus on age, potential disabilities, and environmental factors (e.g., lighting conditions, GPS reception underground).

Removal: When a user decides to stop using a sensor, the sensor will be removed from the home. As removal is the inverse of the initial setup, we decide to combine them with the initial setup in Tables 5.2-5.4.

5.2.5 Example

We illustrate the use of this framework by describing two examples. Both examples are sensors that one might use to detect robbery, which is relevant to when access is granted based on whether there is an emergency in the home. They are also listed in Table 5.2.

Some commercial products, such as the Netatmo Camera [140], alert the user when unrecognized individuals enter the house. As one would expect, cameras and facial recognition algorithms have poor security and privacy qualities, but great usability. They are easily susceptible to replay attacks and adversarial examples. They are also susceptible to physical DoS if the attacker simply blocks the field of vision with an object. Sensor hardware attacks and spoofing are likely impossible for the adversaries we consider. The video stream will capture more information than needed to determine the occurrence of a robbery. Processing the video stream requires long-term cloud storage. Lastly, cameras are ubiquitous and easy to use, although registering users and retraining the facial recognition algorithm to accurately recognize users require some effort.

Glassbreak sensors, like Honeywell's [141], can also detect robbery by monitoring for audio frequencies of glass breaking. These sensors are susceptible to replay attacks, physical DoS, sensor hardware attacks, and spoofing. Machine learning is not necessary, so adversarial examples are not a concern. They capture basic audio frequencies that encode more information than necessary, but this information is simple enough to be stored locally for a short amount of time. They are easy to acquire and use, but they only fulfill the unique purpose of detecting glass breaking. A user looking to sense multiple contexts cannot rely on glassbreak sensors for other contexts.

5.3 Methodology

Both to understand the potential of applying our decision framework in realistic situations and to illustrate how to use it, we applied the framework to sensors that would support commonly desired contextual access control policies in smart homes. Applying the framework requires: (1) a set of desirable contexts for access control policies; (2) sets of sensors that can detect those contexts; and (3) evaluations of the security, privacy, and usability of detecting those contexts with those sensors. This section details our method for applying the framework and analyzing each aspect to create Tables 5.2–5.4.

5.3.1 Desirable contexts

Existing work on context sensing does not fully list the desirable contexts for contextual access control in homes. For example, some work focuses on non-security domains, such as sensing contexts for healthcare [142], activity recognition [143, 144], or indoor tracking [145, 146, 147, 148]. Other work focuses on device-level contexts (i.e., device states) [32, 24, 33, 6], but does not consider contextual access control.

To overcome these challenges, we first identified a list of contexts mentioned in the most closely related work on contextual access control in homes [8, 43, 11]. We then analyzed the user study data from He et al. [8]. We manually clustered participant responses through open coding. We added to our list contexts mentioned at least five times or that are related to identity (thus naturally relating to access control). Tables 5.2–5.4 list the final set of desirable contexts in the leftmost column. The "user" in the leftmost column refers to the initiator of the action who uses a device that is owned by the "owner."

5.3.2 Sensing Mechanisms

Extensive prior work proposes technologies to sense identity or contexts in physical spaces. It is hard for a smart home designer to navigate this work and determine the appropriate sensor based on its security, privacy, and usability trade-offs. For example, to track a person's location in the home, researchers have used cameras [146], CSI (Channel State Information) from WiFi signals [148], visible light channels [149], and more. Direct mappings between contexts and precise sensors are not straightforward. Generally, a physical *sensor* is used to sense some characteristic (which we term an *indicator*) that relates to that context. For example, if age is the relevant context, one might use a person's gait, voice, or facial characteristics as physical indicators of age. These indicators can be sensed with cameras, microphones, and more.

For each context, we identified potential indicators and associated sensors by surveying the sensing literature, searching for relevant industry products, and asking experts from the sensing community for methods they had encountered in their field. Our final set of sensors (see Tables 5.2–5.4) includes both research prototypes and mature products. The *example* column of Tables 5.2–5.4 lists the examples of research prototypes or commercial products we consider for each type of sensor.

To find and evaluate research prototypes, we systematically reviewed the last ten years of proceedings of top conferences in sensing systems (SenSys, MobiSys, and MobiCom), ubiquitous computing (UbiComp/IMWUT), and human-computer interaction (CHI and UIST) in the ACM Digital Library. We first filtered the search results based on keywords ("sensing" in the abstract and "home" in the paper), which yielded 716 papers. We then manually inspected each paper to determine its relevance. We used the paper's title to determine potential relevance, which led to 127 papers remaining. We then read each of these papers to determine its actual relevance. We further excluded papers if (i) they were not related to sensing in homes, but rather applications like VR/AR, smart cities, or health; (ii) they did not focus on sensing a specific context, but rather on refining sensing techniques through improved processing algorithms or machine learning techniques; or (iii) we could not directly map the paper to any of the desirable contexts we identified. The final 36 papers are listed in Table 5.2- 5.4, and we extracted the indicators of the contexts from the corresponding papers. If we did not find prototypes in this body of literature for an indicator, we looked to related top-tier conferences, such as CVPR.

To augment this initial list with more mature and commercially viable methods, we first consulted experts in the sensing community to identify classic papers for types of sensors that are now commonly used. To cover methods used in commercial products, we then searched for sensors of each indicator (as collected from research papers above) on Amazon. If we had not found any indicators at that point for a context, we searched for sensors related to that context and then included the indicators they used. This process led to our final set of

94 pairs of a context that is desirable to sense for access control in the home and a type of sensor (research prototype or commercial product) that identifies that context.

The steps described above survey, but do not systematize, this work. For systematization, we applied our framework to analyze the security, privacy, and usability of using that sensor to detect that context. To understand how the sensing method worked, we read the relevant research papers for prototypes and any user manuals, technical specifications, and white papers we could find for commercial products. We list the detailed criteria we use for this systematization below and in Section 5.2.

5.3.3 Security

Attacks, listed in Section 5.1, target particular types of sensors. To perform replay attacks, one must be able to record and then play back the relevant data, a situation that mostly applies to microphones and cameras. Attacks on sensor hardware target sensors' physical properties and are thus relevant to microphones, MEMS sensors, and more. We used past literature to decide whether the type of sensor used by the sensing method is vulnerable or not.

Some attacks (e.g., physical DoS attacks) are less studied and some sensors (e.g., motion sensors) are less often targeted. In these cases, we studied the sensor's basic principles from papers, product manuals, and white papers, and we discussed among our team whether it might be susceptible to each attack. For example, passive infrared (PIR) motion sensors detect motion based on changes in their view in infrared. Infrared radiation struggles to travel through paper, glass, and thermal blankets, which makes occlusion possible. We acknowledge that some products may adopt anti-tampering techniques not specified in the manual or technical specifications. Our judgments reflect contemplation, rather than lab testing. Tables 5.2-5.4 thus outline expected and potential attacks.

5.3.4 Privacy

We evaluated sensors' privacy implications as follows. We identified the data required by each sensor based on its description in its paper or manual. Examples include audio for microphones, air for smoke detectors, and phone packets for CUPID [150], a WiFi-based indoor localization system. We then identified overprivileged data collection by subtracting the information needed to determine the context from what could reasonably be inferred from the required data. We used the guideline in Section 5.2 to label overprivileged data in Tables 5.2-5.4. For example, Touch ID [151] requires fingerprints. This might suggest overprivilege because a fingerprint is personally identifiable. However, since it is used to detect the user's identity, we do not consider its data collection overprivileged.

Next, we determined the data storage location and retention time required for reasonable performance. For storage location, we examined the algorithms needed to process the data for the sensor. If the sensor required a large amount of longitudinal data or algorithms that could not be computed locally (such as Gaussian models), we labeled the sensor as requiring *cloud* storage. Otherwise, we labeled it as *local*. For example, we consider local storage sufficient for sensors that use SVM classifiers and require only highly limited longitudinal data.

If data did not have to be stored for more than one access, we labeled it *transient*. If any data did, then we labeled it *persistent*. For example, smoke detectors have transient data retention because they do not need to store historical air data to detect future smoke. In contrast, fingerprint readers that verify identity do need to store representations (templates) of the fingerprint to perform future matching algorithms.

5.3.5 Limitations

Due to a lack of access to many of the products and prototypes in our evaluation, the ratings we give are based on team discussion and contemplation. To the best of our ability, we tried to make the criteria as concrete as possible and to review papers and specifications with care. However, some cells in Tables 5.2–5.4 could be subjective and debated by researchers with different assumptions and access to different information. As such, we intend Tables 5.2–5.4 to reflect an initial attempt of applying our framework and distilling the pros and cons of each sensor in each context. We intend these tables as a living document that evolves with community effort and robust online debate.

5.4 Insights From Applying the Framework

We present key findings from applying our framework (Section 5.2) to sensors that support commonly desired contextual access-control policies in smart homes. Tables 5.2–5.4 summarize each sensor's pros and cons in security, privacy, and usability regarding detecting a given context.

5.4.1 Robustness to Attacks

Most sensors are vulnerable to physical DoS. Of the 94 context-sensor pairs evaluated, 64 (68.1%) are vulnerable to physical DoS attacks. Vision-, audio-, heat-, and EM-wave-based sensors (radar, WiFi, radio) can easily be blocked or jammed even by those with no technical background. Vision and heat-based sensors' line of sight can be blocked. Playing loud music floods audio sensors. Energy-absorbent materials can be placed near transmitters (e.g., black material near light-based sensors). Through these means of hindering sensor operation, attackers can become invisible to systems with default-allow policies.

Physical DoS is hard to detect because the symptoms can be similar to normal activities. This is very different from network DoS attacks. Monitoring may alleviate the issue, but home occupants are unlikely to perform constant monitoring. A blocked sensor may not be noticed until the attacker has already achieved their goal.

Sensor redundancy can mitigate physical DoS attacks. For example, a room could have

				Security						**Privacy**				**Usability**						
Contexts	**Indicators**	**Sensor**	**Example**	*Error*	*Replay Attacks*	*Adversarial Examples*	*Physical DoS*	*Sensor Hardware Attacks*	*Spoofing*	*Required Data*	*Overprivileged Data*	*Data Storage*	*Retention Time*	*Wide Availability*	*Initial Set-up / Removal*	*Registration*	*Retraining*	*Reusability*	*Device Dependency*	*Limitations*
User's identity	Voice	Microphone, inertial sensors	[129]	0.1%					!	A,Bm	👎	🏠	○	👎	👍	👍	👍	👎	👎	👎
		Microphone-only	[152]	5-6%						A, C, M	👎	🏠	●	👎	👍	👎	👍	👍	👎	👎
			[113]†	–	!	!	!	!	!	A	👎	☁	○	👍	👍	✋	✋	👍	👍	👎
	Breathing patterns	Microphone	[153]	0.4-2%	?	!				A	👎	🏠	●	👍	👍	✋	👎	👍	👍	👎
	Facial features	Camera	[140]†	Variable	!	!	!			V	👎	☁	●	👍	👍	✋	✋	👍	👍	👍
		Depth camera	[154]†	<0.001%		!	!			P'	👎	🏠	●	👍	👍	✋	✋	✋	👍	👎
		Infrared (IR) camera	[155]†	<0.001%			!			P'	👎	🏠	●	👍	👍	✋	✋	✋	👍	👍
		Camera, inertial, light sensors	[156]	4.7%			!			V, C, E	👎	☁	●	👍	👍	👍	👍	👎	👍	👍
	Eye features	Iris scanner	[157]†	–	!		!			P'	👎	🏠	●	👎	–	✋	👍	👎	👍	👎
	Fingerprint	Fingerprint sensor	[151]†	0.002%	?					F	👍	🏠	●	👎	👍	✋	👍	👎	👍	👎
		Microphone	[158]	2-16%			!	!		A	👎	🏠	●	👍	👍	✋	–	👍	👍	👎
	Body shape	Radar (RF) sensor	[145]	10-21%			!			B	✋	🏠	●	👎	👍	👎	✋	✋	👍	👎
	Bioimpedance	Bioimpedance sensor	[159]	2%						El	👍	🏠	●	👎	👍	✋	👍	👎	👎	👎
			[160]	11-21%						El	👍	🏠	●	👎	✋	👎	–	👎	👍	👎
	Cardiac motion	Radar sensor	[161]	1.39%			!			Bm	✋	🏠	●	👎	👍	✋	👍	👎	👍	👎
		Camera	[162]	1.4-4.5%			!	!		Bm	👎	🏠	●	👍	👍	✋	👍	👍	👍	👎
	Hand gestures	IMU sensors	[163]	10-36.2%			!	!		M	👍	🏠	●	👎	👍	✋	👍	👎	👍	👍
	Gait properties	Vibration sensor	[164]	10%			!			G	✋	🏠	●	👍	👍	✋	–	✋	👎	👎
		Load cells	[147]	7%	?					G	👍	🏠	●	👎	👎	👎	👍	👎	👍	👎
		Pressure sensors	[139]	7.7%	?					G	✋	🏠	●	👎	👎	👎	✋	👎	👍	👎
		Camera	[165]	6.25%	?		!		!	V	👎	🏠	●	👍	👍	👍	✋	👍	👍	👍
		Microphone, WiFi TX & RX	[120]	8%-28%			!			C, A	👎	🏠	●	✋	👍	👎	✋	✋	👍	👎
		Photointerrupters	[166]	1%			!			G	👍	🏠	●	👎	👎	👎	✋	👎	👍	👎
Owner/guest	Identity	*Similar to "Identity" above*		*Similar to "Identity" above*																
User's age	Voice	Microphone	[167]		!	!	!	!	!	A	👎	☁	○	👍	👍	👍	👍	👍	👍	👎
	Facial features	Camera	[168]	6.01 - 6.08 yr.	!	!	!			P	👎	☁	○	👍	👍	👍	👍	👍	👍	👍
			[169]	4.83 - 6.28 yr.	!	!	!			P	👎	☁	○	👍	👍	👍	👍	👍	👍	👍
			[170]	2.514 - 3.086 yr.	!	!	!			P	👎	☁	○	👍	👍	👍	👍	👍	👍	👍
			[171]	22.24 - 9.07%	?	!	!			V	👎	☁	○	👍	👍	👍	👍	👍	👍	👎

Note: In the "Example" column, † denotes commercial sensors or systems.

Table 5.2: An example application of our framework to sensors and contexts identified in our review of the literature and current sensing products. 36 of these sensors come from the academic literature, while the rest are commercial products, denoted with a † in the "Example" column. We mapped the sensors to contexts they are able to detect for the purpose of an access-control policy allowing or denying usage. The "Error" column contains reported values from the cited example sensors. Other columns reflect our best judgment, which was informed by the cited works when related information was reported. !/?/(blank) = Easy/Hard/Impossible, 👍/✋/👎 = Good/Adequate/Poor, 🏠/☁ = Local/Cloud, ○/● = Transient/Persistent data retention, – = Not found. For *Required Data*, **A** = Audio, **B** = Body shape, **Bm** = Body movement, **C** = CSI, **E** = Environment, **El** = Electrical properties of body, **F** = Fingerprint, **G** = Gait, **L/L'** = Geo/Indoor location, **M** = Movement, **P/P'** = Photo/Infrared photo, **D** = Device info, **V/V'** = Video/Infrared video, **T** = Temperature, **O** = Orientation, **Fp** = Floor plan. The rows of this table continue in Table 5.3.

				Security						Privacy				Usability						
Contexts	**Indicators**	**Sensor**	**Example**	*Error*	*Replay Attacks*	*Adversarial Examples*	*Physical DoS*	*Sensor Hardware Attacks*	*Spoofing*	*Required Data*	*Overprivileged Data*	*Data Storage*	*Retention Time*	*Wide Availability*	*Initial Set-up / Removal*	*Registration*	*Retraining*	*Reusability*	*Device Dependency*	*Limitations*
Emergency in the home	Fire	Smoke detector	[172]†	Variable			!		!	E	👍	🏠	○	👍	👎	👍	👍	👎	👍	👍
			[173]†	Variable			!		!	E	👍	🏠	○	👍	👍	👍	👍	👎	👍	👍
		IR Camera	[174]	Variable			!			V'	👎	☁	○	👎	👍	👍	👍	✋	👍	👍
		IR/UV detector	[175]†	Variable			!			E	👍	🏠	○	👎	👎	👍	👍	👍	👍	👍
	Toxic gas	Combustible gas detector	[176]†	Variable						E	👍	🏠	○	👍	👍	👍	👍	👎	👍	👍
		Carbon monoxide detector	[177]†	Variable						E	👍	🏠	○	👍	👎	👍	👍	👎	👍	👍
	Robbery	Camera	[140]†	Low	!	!	!			V	👎	☁	●	👍	👍	✋	✋	👍	👍	👍
		Glassbreak sensor	[141]†	Variable	!		!	!	!	A	👎	🏠	○	✋	👍	👍	👍	👍	👍	👍
User in same *house* as the device	Tag presence	Bluetooth Low Energy (BLE) signal sensor	[178]†	–	!		?		!	L'	✋	🏠	○	✋	👍	✋	👍	✋	👎	👍
		RF/Ultrasonic sensors	[179]	–			!		!	L'	✋	🏠	○	👎	👍	✋	👍	✋	👎	👍
		RFID	[180]	–			!			L'	👍	🏠	●	👍	👍	👍	👍	👎	👍	👍
		WiFi TX & RX	[152]	10%			!		!	A, C, M	👎	🏠	●	👎	👍	👎	👍	👍	👎	👍
	Movement	WiFi TX & RX	[124]	0.5m - 1.1m			!			C	👍	🏠	○	✋	👎	👍	–	👍	👍	👎
			[181]	1.84m			!			C, Fp	👍	🏠	○	✋	👎	✋	–	👍	👍	👎
			[182]	4%			!			C	✋	🏠	●	👍	👍	✋	✋	👍	👍	👎
	Trajectory	Inertial sensors in phones	[183]	1.5 - 2m	!			!		G, M	👎	🏠	●	👍	👍	👎	👍	👍	✋	👍
User in same *room* as the device	Tag presence	BLE signal sensor	[178]†	–	!		?		!	L'	✋	🏠	○	✋	👍	✋	👍	✋	👎	👍
		BLE, IMU sensors	[184]	2.4 - 14.7%			!	!		M, T, O	👍	☁	●	👍	👍	👎	👎	✋	👍	👍
		RF Techniques	[179]	–			!		!	L'	✋	🏠	○	👎	👍	✋	👍	✋	👎	👍
			[185]	0.06%			!			D	👍	🏠	○	👎	👎	👍	👍	✋	👍	👍
		IR tags	[186]	Variable	!		!		!	L'	👎	🏠	○	👎	👎	👍	👍	👎	👎	👍
		Ultrasound TX & RX	[187]	0.1m	!					L'	👎	🏠	●	👍	👎	👍	–	👍	👎	👎
			[188]	3cm	!					L'	✋	🏠	○	👎	✋	👍	👍	👎	👎	👍
		Capacitive NFC	[189]	–						L'	✋	🏠	○	👎	👎	👍	👍	👎	👍	👍
		Visible Light Channel	[190]	5.9cm			!			L'	✋	🏠	○	👎	✋	👍	👍	👍	👎	👍
	Movement	WiFi TX & RX	[124]	0.5 - 1.1m			!			C	👍	🏠	○	✋	👎	👍	–	👍	👍	👎
			[181]	1.84m			!			C, Fp	👍	🏠	○	✋	👎	✋	–	👍	👍	👎
			[182]	4%			!			C	✋	🏠	●	👍	👍	✋	✋	👍	👍	👎
		Motion sensor	[180]	0.5 - 1.1m			!			M	👍	🏠	●	👍	👍	👍	👍	👎	👍	👍
			[191]†	1.84m			!	!		M	👍	🏠	○	👎	👍	👍	👍	✋	👍	👍
	EMI	Voltage sampling	[192]	6%						L'	✋	🏠	○	👎	✋	👍	👍	👎	👎	👍
		Passive magneto-inductive sensors	[193]	6-17.4%						L'	✋	🏠	○	👎	👍	👎	👍	👎	👎	👍
	RF reflection	RF sensor	[123]	81%			!			L'	👎	☁	●	👎	👍	✋	✋	✋	👍	👍
	Electric potential	Electrical potential sensors	[194]	0.16m			!			El	👎	🏠	○	👎	👍	✋	👍	👎	👍	👍
	Location semantic	WiFi, microphone, IMU sensors, Barometer	[185]	0.63-0.78			!			L'	✋	🏠	○	👍	👍	👍	👍	👍	👎	👍
	Hand gestures	IMU sensor	[163]	10 - 15%				!		L'	✋	🏠	●	👍	👍	👎	–	✋	👎	👎
	Water pressure	Pressure sensor	[195]	17.3 - 29.9%						L'	✋	☁	●	👎	👎	👎	–	👎	👍	👍
Owner away or not	Location	GPS	[196]†	Variable			!		!	L	👎	☁	●	👍	👍	👍	👍	👍	👎	👎
Adult nearby	Age	*Similar to "Age" above*		*Similar to "Age" above*																

Note: In the "Example" column, † denotes commercial sensors or systems.

Table 5.3: A continuation of the rows of Table 5.2, which is an example application of our framework to the sensors and their associated target contexts. The abbreviations used are the same as defined in Table 5.2's caption.

Contexts	Indicators	Sensor	Example	Security						Privacy				Usability						
				Error	*Replay Attacks*	*Adversarial Examples*	*Physical DoS*	*Sensor Hardware Attacks*	*Spoofing*	*Required Data*	*Overprivileged Data*	*Data Storage*	*Retention Time*	*Wide Availability*	*Initial Set-up / Removal*	*Registration*	*Retraining*	*Reusability*	*Device Dependency*	*Limitations*
No one nearby	WiFi signals	WiFi TX & RX	[182]	96% (TPR)			!			C	👍	🏠	○	👍	👍	👉	👉	👍	👍	👎
			[197]	Low	!				!	D	👉	🏠	●	👉	👍	👍	👍	👉	👎	👎
	Presence	RF sensor	[145]	High	?		!			B	👉	🏠	●	👎	👍	👎	👉	👉	👍	👎
		Camera with IR	[198]†	Variable			!			V'	👎	☁️	●	👍	👍	👍	👍	👍	👍	👍
			[199]	Variable						V'	👎	☁️	●	👎	👎	👉	👎	👉	👍	👍
		Load cells	[147]	7%						G	👍	🏠	●	👎	👎	👎	👍	👎	👍	👎
		Pressure sensors	[139]	7.7%						G	👉	🏠	●	👎	👎	👎	👍	👎	👍	👎
		Ultrasonic sensors	[200]	10%			!			B	👉	🏠	●	👎	👍	👎	👍	👉	👍	👎
	Movement	Motion sensor	[201]†	Variable			!			M	👍	🏠	○	👎	👍	👍	👍	👉	👍	👍
			[202]†	Variable			!	!		M	👍	🏠	○	👎	👎	👍	👍	👎	👍	👍
			[203]†	Variable			!	!		M	👍	🏠	○	👉	👉	👍	👍	👉	👍	👍
	Footsteps	Microphones		Variable	?			!	!	A	👎	🏠	●	👍	👍	👍	👍	👍	👍	👍
		Similar to "Gait" above						*Similar to "Gait" above*												
	CO$_2$	Nondispersive Infrared (NDIR) CO$_2$ sensors	[204]†	Variable						E	👉	🏠	○	👉	👍	👍	👉	👎	👍	👍
	Body heat	Infrared sensors	[205]†	Variable			!			V'	👎	🏠	○	👎	👍	👍	👍	👉	👍	👍
People asleep nearby	Movement	Inertial sensors	[206]†	Variable				!		M	👉	🏠	○	👉	👍	👉	👍	👍	👎	👍
			[207]†	Variable				!		M	👉	🏠	○	👍	👍	👉	👍	👍	👎	👍
		Similar to "Motion sensors" above						*Similar to "Motion sensors" above*												
		Radar sensor	[208]	89.6% (recall)			!			M	👎	🏠	○	👎	👍	👍	👍	👎	👍	👍
People present in same *house* as the user	Location	GPS	[196]†	Variable			!		!	L	👎	☁️	●	👍	👍	👍	👍	👍	👎	👎
	Movement	Static electrical field	[209]	1.88%			!			E	👉	🏠	●	👎	👍	👎	–	👎	👎	👍
		RF sensors	[122]	Low			!			M	👎	☁️	●	👎	👍	👎	👍	👉	👎	👍
	Tag presence	RF/Ultrasonic sensors	[179]	–			!		!	L'	👉	🏠	○	👎	👍	👍	👍	👉	👎	👍
		BLE signal sensor	[178]†	–	!		?		!	L'	👉	🏠	○	👉	👍	👉	👍	👉	👎	👍
People present in same *room* as the user	WiFi signals	WiFi TX & RX	[210]	Variable			!			L'	👎	🏠	○	👍	👉	👉	–	👍	👍	👍
			[150]	1.8m	!		!			C	👉	🏠	●	👉	👍	👉	👍	👉	👎	👍
	RF reflection	RF/Ultrasonic sensors	[123]	19%			!			L'	👎	☁️	●	👎	👍	👉	👉	👉	👍	👍
	Sound (chat)	RF/Ultrasonic sensors	[152]	26%			!	!		L'	👎	🏠	●	👎	👍	👎	👍	👍	👎	👎
	Doorway activity	RF/Ultrasonic sensors	[200]	10%			!			B	👉	🏠	●	👎	👍	👍	👍	👉	👍	👎
	BLE signals	BLE signal sensor	[178]†	–	!		?		!	L'	👉	🏠	○	👉	👍	👍	👍	👉	👎	👍

Note: In the "Example" column, † denotes commercial sensors or systems.

Table 5.4: A continuation of the rows of Table 5.3, which is an example application of our framework to the sensors and their associated target contexts. The abbreviations used are the same as defined in Table 5.3's caption.

a motion sensor, a pressure sensor in the floor, and a microphone to detect whether the room is occupied or not. If access is granted when the room is unoccupied, an attacker wanting access would need to accomplish the difficult task of occluding all three sensors around the same time. By cross-checking the sensors' data streams with each other [44], the system could verify whether the room is unoccupied and determine whether a sensor has been compromised.

Careful policy design is another defense against physical DoS attacks. A system's default policy—whether to *allow* or *deny* access when a condition is met—can impact attack success. For example, a user might specify "my child should not have access to the TV." With a default-allow policy, TV access will be granted unless a child is detected, yet the child can block a sensor to avoid detection. With a default-deny policy, the child cannot rely on physical DoS.

The optimal default policy may vary based on the device or operation. Users may prefer default-allow rules for controlling lights because falsely allowing operation is typically of little consequence, but falsely denying operation causes inconvenience [8]. A sensor's false positive/negative rates also play a role. Smart home designers should help users navigate these nuances through sensible default policies and templates.

Many sensors are susceptible to physical DoS attacks. Mitigations against physical DoS of sensors include redundant sensors of different types and carefully constructed default policies.

Audio- and vision-based sensing is vulnerable to many attacks. Basic audio-based sensing is susceptible to all types of attacks in Tables 5.2-5.4 [19, 131, 132, 20]. Visible-light camera sensing is also susceptible to all of these attacks, except for hardware attacks. For cameras, spoofing can be difficult, but replay attacks with photo or video input are feasible.

Existing defenses for sensing methods are insufficient for access control because they were designed for *authentication* instead. Most prior work on audio- and camera-based sensing lacks security analyses. The few that analyzed security focused on replay and spoofing attacks. Authentication assumes that unrecognized users are unauthorized. Thus, a large body of research has focused on preventing replay and spoofing attacks against audio- and camera-based sensing to avoid attackers from becoming recognized in this regard. A commonly proposed defense is to rely on secondary channels of information on the same device [211] or other devices [100, 44]. For example, 3D cameras (like Face ID on iPhones [154]) analyze depth information to deter simple, photo-based replay attacks. However, in access control, default-allow policies authorize *unrecognized* users, resulting in the possibility of physical DoS attacks. Therefore, for such policies, an attacker can gain access by targeting one information channel (e.g., targeting an image's visual features by presenting a photo) and becoming unrecognizable to the system.

Existing defenses for audio- and camera-based sensing focus on attacks that compromise authentication, not access control. Attackers can exploit the default semantics of access-control policies to gain access, and physical DoS attacks become easier.

Physical adversarial examples can be effective for skilled, external attackers. For sensing methods that rely on machine learning, we noted whether they were susceptible to adversarial examples. Specifically, within the scope of context sensing and our threat model, we consider only physical adversarial examples. The attacker misleads the algorithms by adding physical perturbations to the environment or to themselves, instead of feeding data to the algorithms directly. Recent work has demonstrated the feasibility of such attacks for images [18, 95, 212] and audio [132, 131, 20]. Although some attacks require whitebox access to models, which is unrealistic for commodity smart home devices, blackbox attacks are also possible [212, 213, 214, 215, 216].

Internal attackers are less likely to use physical adversarial examples because they require substantial technical skills and resources to generate and test. Instead, they would use familiarity with the system to launch replay, spoofing, or physical DoS attacks to a similar end. However, if we consider *external opportunistic* attackers (e.g., a group of burglars) who do not have information about the victim, physical adversarial examples can be very effective. In fact, untargeted adversarial examples are strictly easier than targeted attacks. For example, attackers might want to attack face recognition on all security cameras in a neighborhood. In doing so, they can reuse and refine their adversarial examples.

Internal attackers may prefer replay, spoofing, and physical DoS attacks. Opportunistic external attackers may prefer adversarial examples.

5.4.2 Privacy

Except for cameras, cloud storage is not usually required when sensing contexts. We found that 79.8% ($n = 75$) of the examined sensing techniques do not require data storage on the cloud. Unfortunately, 10 of the 14 methods that use cameras do require cloud processing. Oftentimes, cloud storage is necessary for computationally intensive algorithms or large training datasets required to process video or image data online (e.g., neural networks for facial recognition). Privacy-preserving machine learning may alleviate this need. One approach is to protect the privacy of the training data. In federated learning [217], sensitive data stays local and only gradient updates are sent to the server. Another approach targets the inference stage by running the models locally or on the edge [218, 219]. Companies may prefer cloud storage because they can collect user data. Despite the risk of data exposure, some users may prefer cloud storage if it costs less.

Few sensing methods, often camera-based ones, require cloud processing. Federated learning or performing ML on the edge could obviate cloud processing.

Cameras/microphones are invasive but currently indispensable, thus necessitating privacy countermeasures. Users perceive age to be an important context for access control [8]. Unfortunately, most existing age-estimation methods rely on cameras or microphones, raising privacy concerns. Until privacy-preserving methods for age detection become possible, users may instead wish to record age while registering their identity during system setup.

Suppose cameras and microphones have to be used. To enhance bystanders' privacy, countermeasures against these sensing methods have been proposed, such as strategically blurring an image or jamming microphones with ultrasonic noise [64, 21, 22]. These proposals improve privacy, but also imperil the access control system, making it more likely to ignore attackers or confuse attackers with benign users. Therefore, detecting contexts with obfuscated sensor data may be another research direction. Raval et al. [220] proposed a utility-aware obfuscation mechanism for smartphone apps, which shows a promising road to privacy-preserving sensing in homes.

Privacy-invasive sensors may be essential. Privacy protections may weaken the access-control system.

Mismatch between required and collected data. Only 25 of 94 context-sensor pairs (26.6%) do not collect more data than needed to deduce the context. In contrast, 33.0% were *acceptable* and 40.4% were *poor* in our analysis. Most sensing methods marked as *poor* record unnecessary video or audio. Manufacturers typically rely on high-fidelity sensors, such as cameras or microphones, to sense contexts. This also happens when researchers use microphones on voice assistants or smartphones for ultrasonic-based sensing for their *wide availability*. While federated learning or edge computing may mitigate privacy concerns, they may also appear cryptic to the average user. These methods may therefore fail to alleviate user concerns about sensors inadvertently collecting invasive data. Future work should investigate effective means of communicating to users privacy considerations, such as using privacy labels [221] or visual indicators [222].

Competing interests between multiple stakeholders—manufacturers, researchers, designers, users—also contribute to this mismatch between the data required and the data collected.

The designer might only want to know which room the user is occupying, but manufacturers and UbiComp researchers likely would want to collect information about the activity of the user in that room. Obtaining this extra knowledge enables the latter two parties to design and provide technology benefiting users in other aspects of their daily life. For the benefit of smart home owners and users, smart home systems and sensors should offer the ability to prioritize utility or privacy.

Most sensors collect more data than needed. User awareness and control of data collection is critical.

5.4.3 Access, Deployment, and Acceptability

Many sensing methods for authentication are not inclusive. Research in sensing and access control is generally not inclusive to the elderly and groups with various disabilities. For example, the gait-sensing literature mostly does not consider people with walking disabilities. For inclusivity, contextual access-control systems must offer an array of sensors that allow *every* individual to authenticate an identity or person-specific context.

5.5 Conclusion

Contextual access control in homes is desirable, yet mostly unsupported. To bridge this gap, sensors can be used to detect contexts. However, they must defend against both expert and non-expert adversaries while respecting user privacy and usability. We proposed both a new adversarial model for context sensing in homes and a decision framework for evaluating potential sensors in terms of security, privacy, and usability. We applied this framework to common sensors through literature systematization, finding important trade-offs. We have made our framework and evaluations accessible in a public GitHub repository to facilitate updates and public discussion.

CHAPTER 6
AUTOMATICALLY GENERATING GENERALIZABLE NETWORK ALLOWLISTS FOR HOME IOT DEVICES

In addition to software and hardware, home IoT devices also have suffered from numerous network attacks [7]. Attackers have exploited vulnerabilities in home IoT devices' software, protocols, and default settings to compromise devices [223] for purposes including creating botnets like Mirai [224] and Hajime [225]. Contributing to security issues is the array of (often inexperienced) manufacturers creating IoT devices, the difficulties of deploying software patches to devices without screens or traditional user interfaces, and the lack of standardization [5].

Rather than relying only on potentially unresponsive vendors to patch devices, households could control the network traffic of their own IoT devices, applying security policies designed to minimize problematic network behaviors like distributed denial of service (*DDoS*) attacks or the exfiltration of data about the home to illegitimate endpoints. Specifically, the household could use firewall-style rules to limit which hosts a particular device may contact. Recent papers [69, 226, 227] have employed *blocklists*, which specify (problematic) hosts that cannot be contacted and permit traffic to all other hosts. From a security perspective, an even more attractive approach would be to employ *allowlists*, which instead enumerate the hosts that can be contacted and block traffic to all other hosts. While allowlist-based approaches have a much smaller attack surface, they are rarely used in practice because enumerating the destinations that general-purpose computing devices should be able to contact is typically intractable. In contrast, many home IoT devices' actions are highly limited. Intuition thus suggests that allowlists may be practical for securing home IoT devices at the network level. A key challenge, however, is that allowlists (unlike blocklists) must be complete for devices to function correctly.

Some prior papers have, like us, proposed allowlists for IoT devices in one form or another [71, 72, 70, 73]. These prior papers, however, created allowlists and then subsequently evaluated those allowlists on individual devices in homogeneous settings (e.g., generating an allowlist from a single Nest Thermostat in a lab and then evaluating that allowlist on that same device in the same lab).

In this chapter, we instead explore how to create allowlists for home IoT devices that generalize and transfer across settings and instances of devices. Studying a single device in a controlled environment, as in past work, misses key sources of variability across devices, regions, network architectures, and time. Alternatively, requiring each user to create a personalized allowlist can require an onerous time investment for users who would need to exercise all of a device's functionalities. Furthermore, creating an allowlist from a single device that has already been compromised would nullify any security guarantees. To study the design space for generalizable allowlists, we conduct a number of experiments using data shared with us by the IoT Inspector project [26], which collected network traffic data from thousands of IoT devices deployed in real homes. This approach lets us analyze how the network behaviors of a single *product* (e.g., the Philips Hue lightbulb) varies across *devices* (e.g., a single household's physical instance of a Philips Hue) in different homes across the world.

As our first contribution, we characterize variability in the network behaviors of distinct devices of the same product (e.g., Philips Hue lightbulbs from different households), the first step in mapping the design space for allowlists. We used the IoT Inspector dataset to compare and contrast the network behaviors of 20 popular IoT products across households (Section 6.3). We focused on the hostnames (i.e., fully qualified domain names) and ports with which each device communicates. We expected to see variations based on network-specific configurations (e.g., DNS and NTP), which we indeed observed. To our surprise, we also observed more substantial variation in the hostnames contacted by different devices of the same product beyond configuration-specific facets. On average, 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. Nonetheless, many hostnames in this "long tail" were related to each other. Some hostnames contained what appeared to be randomized substrings (e.g., `oculus2975-us1.dropcam.com` vs. `oculus1802-us1.dropcam.com`) presumably to support load balancing. Others contained apparent regional identifiers (e.g., `avs-alexa-6-na.amazon.com` vs. `avs-alexa-7-eu.amazon.com`). As a result, an allowlist of hostnames created by analyzing the network traffic of one specific device is surprisingly unlikely to generalize to other devices of that same product.

To enable more nuanced allowlists, **our second contribution is thus an evaluation of the design space for allowlists, showcasing how different designs impact both the fraction of observed traffic allowed and the attack surface** (Section 6.4). First, we compare different representations of hosts. Compared to an allowlist of hostnames, an allowlist of second-level domains is far more likely to capture legitimate variability and thus generalize. Unfortunately, many IoT devices rely on cloud services (e.g., AWS), meaning that domain-based allowlists open a large attack surface. As a compromise, we propose and evaluate a pattern-based representation of hostnames, which appears to enable transferring allowlists across devices while limiting the attack surface. Similarly, to make allowlists more robust in the presence of a small number of mislabeled or compromised devices, we explore requiring that a host be contacted by n different devices (termed the **threshold**) for it to be added to the allowlist. We also investigate the effect of the number of different devices of a product necessary to create a robust allowlist, finding it to be on the order of one or two dozen devices, depending on the host representation.

To evaluate whether allowlists generated from the two-year-old IoT Inspector dataset would generalize, **our third contribution entailed enforcing these allowlists on real devices in our lab, analyzing how they impacted device functionality.** The aforementioned experiments were retrospective simulations on the IoT Inspector dataset, raising the question of whether the allowlists created would actually work in practice in a different setting. Thus, Section 6.5 reports on an in-lab experiment in which we tested nine popular home IoT products' range of functionality when various types of allowlists generated from IoT Inspector were enforced. While we found that some actions (e.g., streaming music, downloading apps) occasionally failed for some products, most other actions we tested worked normally under the allowlists tested. While we had hypothesized that rare actions (e.g., reinitializing the device) might fail because they likely had not been captured in IoT Inspector, we observed that these rare actions typically functioned normally. In other words, we found that snapshots of two-year-old network traffic data from other households' devices can be sufficient for generating allowlists for home IoT devices if the allowlists themselves are designed with sufficient care and nuance.

6.1 Problem Setting and Terminology

In this chapter, we create our threat model, where a remote attacker aims to compromise these IoT devices, forcing them either to contact a server under the attacker's control (e.g., for data exfiltration or botnet command and control) or to direct traffic toward some victim (e.g., as part of a DDoS attack). The attacker always has the ability to launch attacks from a new server. However, they do not have the ability to compromise the IoT device 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.

The user (defender) aims to create an **allowlist**, or specification of the endpoints with which a given home IoT device may communicate. For instance, an allowlist might specify that a given home IoT device can only communicate with `meethue.com`; all other traffic, both inbound and outbound, is blocked. In a given home, different products will likely have different allowlists, but multiple devices in the home of the same product (e.g., all six of the user's Wyze Cameras) will share an allowlist. We assume the contents of the allowlists are known to both the defender and the attacker.

We create allowlists based on a large dataset of IoT network traffic from many devices (see Section 6.2). We test the allowlists generated on both held-out data from the same dataset (Section 6.4) and on real devices in our lab (Section 6.5).

In describing the allowlist creation process, we use the following terms. **Vendor** refers to a company that makes home IoT products (e.g., Amazon). **Product** refers to the collection of devices sold under a specific name (e.g., Amazon Ring), potentially encompassing multiple releases and versions. **Device** refers to a single physical object that is an instance of a home IoT product (e.g., a single Amazon Ring in a home is a single device). **Type** refers to the category of multiple home IoT products with similar purposes (e.g., both the Amazon Ring and Wyze Camera are of type "home IoT camera").

We analyze three main design dimensions for allowlists. First, we compare **host representations**, or abstractions for characterizing endpoints. These representations include lists of hostnames, domains, IP addresses, and more (Section 6.4.3). Second, we compare **thresholds**; for a threshold of n, we only add a particular host (per the host representation being tested) to an allowlist if it appears in the traffic of at least n devices in the sample. Thresholds aim to make allowlists more robust to noise, mislabeled data, and potentially a small number ($< n$) of compromised devices. Third, we compare **sample sizes**, or the number of unique devices of a single product (from different households) used to generate an allowlist.

6.2 Dataset

The dataset we use to analyze the variability of devices' network traffic (Section 6.3), generate allowlists (Sections 6.4–6.5), and retrospectively analyze the impact of different allowlist design decisions through simulations (Section 6.4) is the IoT Inspector dataset [26], which the authors of that work shared with us with the approval of their IRB. Specifically, we analyze the subset of IoT Inspector containing only devices with validated product names and network traffic recorded. This subset consists of 5,439 unique devices, representing 424 unique products from 80 different vendors among 43 device types. These 5,439 unique devices came from 1,468 households in total. Original data collection occurred between April 8 and July 24, 2019, documenting 38,164,870 network flows in total. The authors of that work define a network flow as the combination of the device ID, local port, and remote IP address and port [26]. To improve clarity, in the body of this chapter we report on the 20 products (3,456 devices total) that appeared most frequently in the dataset. The appendix presents analogous graphs with the 52 products for which there was enough data to perform our analyses.

The dataset contains the following information, some of which involved post-processing by the original IoT Inspector authors. Each flow in the dataset contains the **product and vendor** of the home IoT device whose traffic was being monitored; this information was entered manually by participants on the IoT Inspector user interface. Each flow 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 [26] that included examining DNS traffic. They attempted to associate the remote IP address with a **hostname** (fully-qualified domain name) and **domain** (top-level and second-level domain, such as `google.com`). Each flow is also associated with the household's **timezone**, as opposed to a more precise geo-location, to respect participant privacy.

As with any real-world dataset, data cleaning was necessary. While the original authors performed some data cleaning [26], 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 other devices with the same product name commonly contact. We considered a domain to be common if at least 20% of devices for the product have contacted it, an arbitrary value we chose to reduce the likelihood of mislabeled devices. In Section 6.4, we use thresholds to further account for mislabeled devices. We made an exception for devices that only contact DNS or mDNS resolvers because these may be devices that are simply not used during the data collection period. We found 16 potentially mislabeled devices across 10 products, and we excluded them from further analysis. We manually examined other outliers. We found one device, labeled as a Belkin Wemo switch, that contacted 26,594 unique domains, which was very unusual relative to other Belkin Wemo switches. We suspected this one device might be compromised or mislabeled, 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. 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 one, then we concluded that it had the correct hostname. To mitigate the effects of virtual hosting, whereby different domains can map to the same IP address, 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 instead of IP addresses. This step left us with 30.4% of flows with an unreliable hostname. We kept these unreliable hostnames to avoid missing data, yet retain an annotation about their unreliability.

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. For example, 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 some traditional analyses, such as identifying patterns in inter-packet arrival times. However, these privacy-preserving limitations would likely apply to any crowdsourced dataset, and we find that we are nonetheless able to generate allowlists that generalize to our lab.

Availability We have made our code for conducting the analyses reported in this chapter available in a GitHub repository [228]. We also uploaded the code to hotcrp as "additional materials." Unfortunately, the IoT Inspector dataset cannot be shared without explicit permission from the original authors' IRB [26]. We instead provide synthetic sample data in the correct format.

6.3 Variability of Device Behavior

Ideally, a user would have complete knowledge about a product's behavior when creating an allowlist. Unfortunately, vendors typically do not provide that information. With the help of the IoT Inspector dataset, we instead use observed network traffic from one or more devices of a given product to create potential allowlists. However, this approach of transferring observations of one device's network behaviors to another device of the same product (ostensibly owned by a different person and on a different network) will only work if the allowlist can capture differences in network traffic. Thus, in this section, we use the IoT Inspector dataset to analyze how devices of the same product vary in network behavior. We

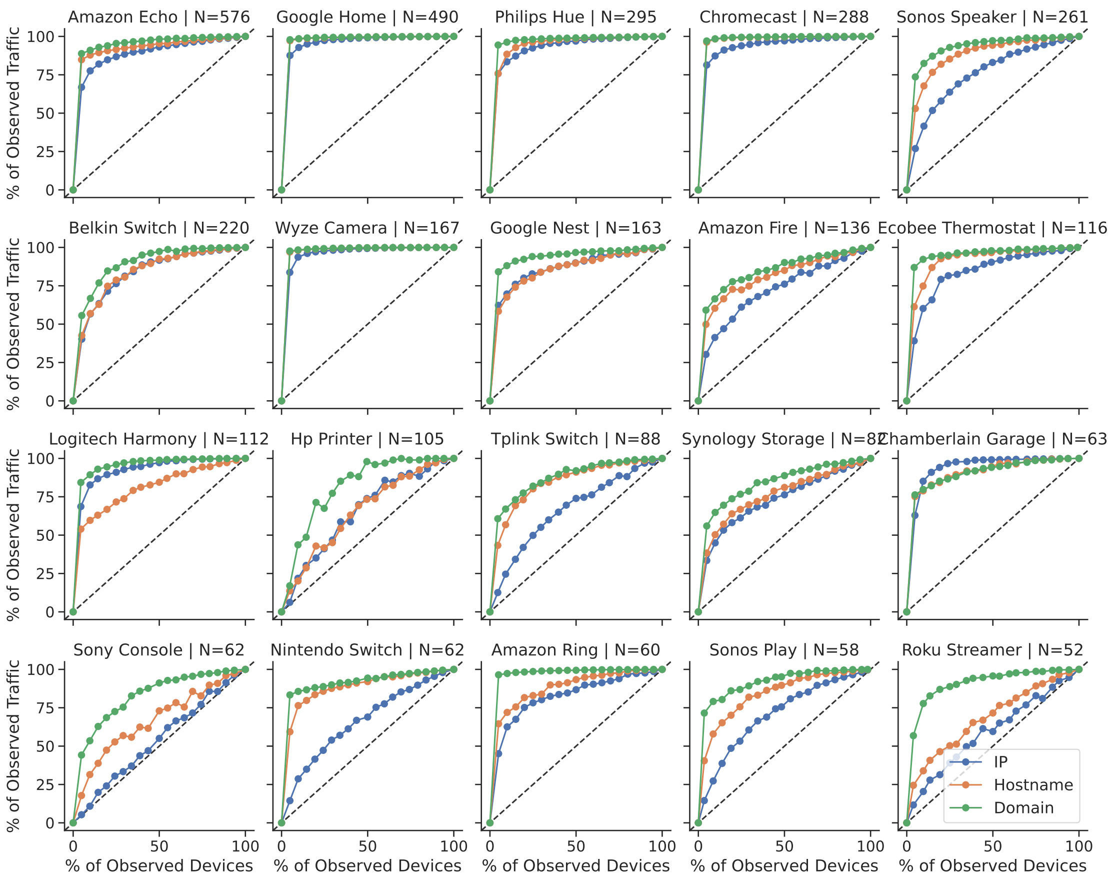


Figure 6.1: Flow coverage (averaged across 100 runs) for three ways of characterizing endpoints (domain, hostname, IP address) for the 20 most popular products. Coverage well below 100% for small samples indicates high variability across devices.

find substantial variation for reasons including load balancing, under-represented regional services, and user-defined DNS resolvers.

6.3.1 Coverage of Network Behaviors

As a first step, we took the union of the hosts seen across all devices for each of the 20 most common products in the IoT Inspector dataset, analyzing what fraction of those hosts were observed in samples of devices for that product. We use the term **coverage** to refer to the percentage of flows (remote host and remote port) contacted by any device of that product that were contacted by a given sample of those devices. If devices behaved similarly to each other, coverage would approach 100% for even a small sample.

For a number of products, however, this was not the case. Figure 6.1 shows the coverage of different sample sizes for three different 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 average the coverage over 100 random samples. For most products, the coverage of domains approaches 100% even for small samples. However, the coverage for hostnames and IP addresses was often substantially lower than for domains at a given sample size. This result suggests that, for many products, a given device may be contacting different hostnames and IP addresses, even when it is contacting the same domain, creating challenges for more granular allowlists.

While we expected to observe lower coverage for IP addresses, we were more surprised that hostnames followed a similar pattern for some products (e.g., Google Nest and Amazon Ring). Averaged across all 20 products, coverage of 95% of flows was only achieved with a sample of 58% of the devices of that product in the dataset when using hostnames, compared to 67% when using IP addresses. For 99% coverage, the percentages increase to 81% (hostname) and 88% (IP address). The low coverage observed with small samples supports our assumption that generating an allowlist from a single device — as done by prior work [71, 72, 70] — is likely to miss important variability and thus fail to create generalizable allowlists, particularly for allowlists based on hostnames or IP addresses.

6.3.2 Sources of Variability

Creating generalizable allowlists that transfer across devices requires further understanding potential root causes of variability.

Load Balancing and Caching One of the main sources of variability appears to be the use of different, but related, hostnames to support load balancing and caching. For example, Apple Push Notification (APN) service sets up multiple servers for better availability, using hostnames like `27-courier.push.apple.com`, where the number (27) varied across devices in our dataset. When a large number of similar hostnames exhibit small variations, an allowlist created based on precise hostnames is unlikely to generalize.

The good news is that most of these hostnames follow patterns. In the simple example above, the number is the only part of the the hostname we observed changing, which suggests a more abstract hostname representation (see Section 6.4.3) would better generalize. Some other 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.

Regionalization Geographic bias in IoT Inspector, collected by US-based researchers using a tool documented in English, is another consideration. In the subset of IoT Inspector we analyzed, 70.1% of devices appear to be in North or South America based on timezone.

However, we observed that devices in different geographic regions sometimes contact different endpoints. For example, in North America ("NA") Amazon Echo tended to contact hosts like `avs-alexa-6-na.amazon.com`, whereas those in Europe ("EU") tended to contact hosts like `avs-alexa-7-eu.amazon.com`. Some variability was more subtle than simply replacing region codes. For example, Belkin Wemo switches in North and South America typically contacted `navy.mil` for network time protocol (NTP) services, while those in other regions usually contacted `tu-berlin.de` instead.

The streaming media destinations that users visit also vary across regions, reflecting local interests and making allowlists less likely to generalize across regions. For example, we observed endpoints like `tf1.fr` (a French TV channel contacted by a Google Chromecast), `bell.ca` (an ISP in Canada contacted by an Amazon Echo and a Chromecast), and `met.no` (a Norwegian weather service contacted by a Synology network storage device to display weather information), highlighting how geographic biases impact allowlists.

DNS Because users (or their ISPs) typically assign a local DNS resolver, DNS traffic was an additional (expected) source of variability. We observed a long tail of DNS traffic, including 102 unique DNS resolvers contacted by only one device.

Variable Remote Ports Although most remote hosts used only a few dedicated ports (e.g., port 443 for HTTPS), as shown in Figure 6.4, some remote hosts surprisingly used seemingly random remote ports. For example, the 82 Synology storage devices in the dataset contacted 45,963 ports on remote hosts. Synology documents that the ports used for data connections in "Passive Mode" can range from 1025 to 65535 [229]. Similarly, while Amazon Ring devices in our dataset contacted only 72 unique domains, 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. Blocking ports on these devices may break their functionality.

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. Similarly, two Amazon Echo devices from another household contacted `sbcglobal.net` through 3,858 remote ports. Both domains belong to AT&T services [230].

Other Causes Some variability may be artifacts of IoT Inspector's data collection process. 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 sending out DNS queries. Prior work [231] also found, for instance, that Belkin Wemo plugs transmitted IP addresses in payloads at the time of their data collection in 2020, though our experiments found they now transmit hostnames.

Limitations Some products (e.g., Amazon Echo) have multiple generations. We grouped these generations into a single product. It is possible that different generations may behave differently. Even for the same generation, behavior may change depending on the software or firmware version. As we detail in Section 6.5, a recent update to Belkin Wemo smart plugs caused them to contact a completely different set of endpoints. Furthermore, product names were provided manually and some volunteers gave only a general name (e.g., "Sonos speaker"), potentially collapsing different models.

6.4 Generalizability of Allowlists

To investigate how the design of allowlists for home IoT devices impacts the network traffic permitted, we must first devise a method for producing allowlists that can reflect various design factors. We develop and evaluate a proof-of-concept approach that generates allowlists, varying the host representation (granularity and abstraction of endpoints), thresholds, training samples, and more.

6.4.1 Methods

We introduce five variables for allowlist generation. **Host representation** describes the level of abstraction for describing hosts. In addition to our focus on domains, hostnames, and IP addresses in the body of the chapter, the appendix also compares (less successful) formats, such as subnet or BGP prefix. **Ports** can be added on top of any selected selected host representation. We discuss their impact independently. As previously defined, **threshold** is an integer specifying the minimum number of devices on which we need to observe an endpoint to add it to the allowlist. **Training data** references the data used to generate the allowlist; it can be chosen based on criteria like the product or timezone. **Testing data** references the dataset used to evaluate the generated allowlist. A single device is never in both the training and testing data for an experiment.

As further discussed in Section 6.4.3, one hostname representation we consider, the **pattern** representation, provides a partial abstraction of the hostname. To generate these patterns, which are effectively regular expressions, we use DBSCAN to group similar hostnames for each product. We choose DBSCAN because we do not know the number of clusters in advance and would like to recognize unique hostnames that should not be clustered. We cluster each hostname based on its lowest-level subdomain (i.e., the leftmost part of the FQDN) and use the remaining part of the FQDN for grouping so that different domains or subdomains are not clustered together. We condense any consecutive numbers to "[0-9]+" because digits rarely carry any meaning in this context. Once a cluster has been identified, we generate an ordered list of the longest non-overlapping common substrings for all hostnames in the cluster. If we can further group the remaining parts, we accept these variations rather than using a wildcard to minimize the attack surface. If the remaining parts appear random, a wildcard replaces them. For example, hostnames used by Apple Notification Services become `[0-9]+-courier.push.apple.com`. Spotify access-point hostnames become `guc3-accesspoint-a-.*.ap.spotify.com`.

6.4.2 Key Metric

We evaluate potential allowlist design choices 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 allowed flows (MFAF)**. Intuitively, the MFAF captures the fraction of traffic previously observed (without the allowlist) that would still have been permitted with the allowlist active. A high MFAF is desirable since it reflects the device's observed traffic proceeding as normal. We take the median fraction instead of the mean because the distribution can be very skewed. Formally, MFAF is defined:

$$MFAF(A_D, D') = median(\{\frac{|F^d_{A_D}|}{|F^d|}, \forall d \in D'\}) \tag{6.1}$$

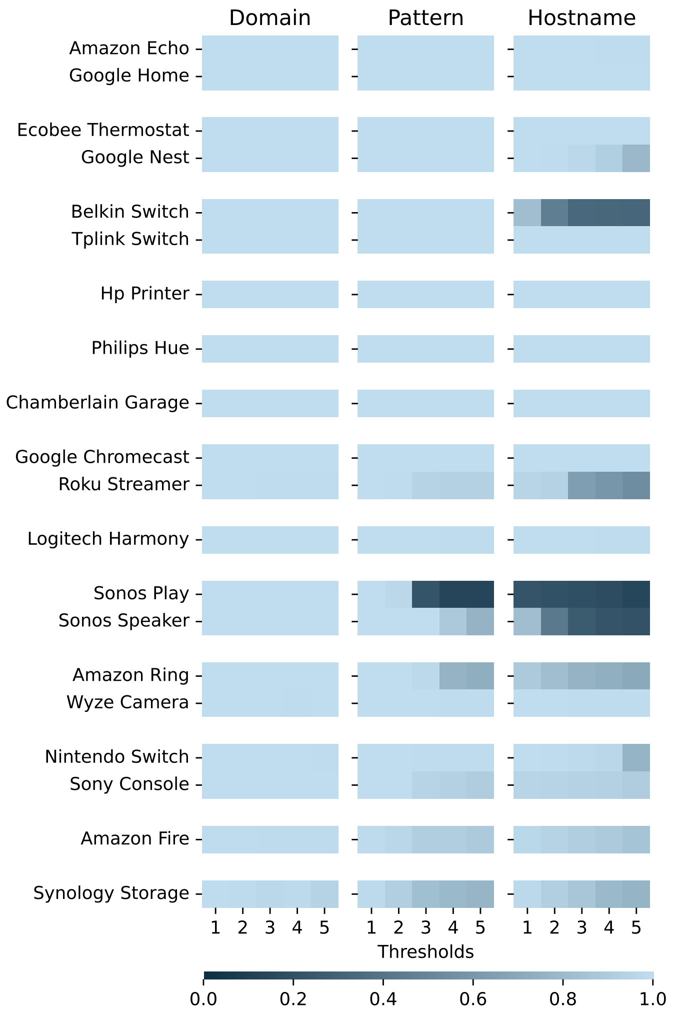


Figure 6.2: MFAF of allowlists generated from, and applied to, the same product in a train-test split. The products (rows) are grouped by type. The color indicates the proportion of flows permitted using a threshold of between 1 and 5.

where D is the group of devices used for generating allowlist A_D, D' is the group of devices that the allowlist is applied to, d is a device in D', F^d is the all the flows transmitted on d, and $F^d_{A_D}$ is the fraction of d's flows allowed when the allowlist A_D is active. We require $D \cap D' = \emptyset$ for a fair evaluation. A flow is identified by a combination of the device ID, the local port, the IP address of the remote endpoint, and the remote port.

6.4.3 Host Representation

We tested seven possible host representations. In this section, we contrast representing endpoints by their domain, pattern, and exact hostname. Appendix 7.9 presents the four less successful alternatives.

As shown in Figure 6.2, all 20 popular devices had high MFAF (greater than 0.9) when using domain representations. While this means that the (presumably legitimate) traffic observed in our test set would be allowed, domain-based allowlists also have the largest attack surface, as we revisit later in this section.

In contrast, both hostname and pattern representations, particularly the latter, seemed to better balance generalizability and security. With the threshold set to 1, hostname representations achieved an MFAF greater than 0.9 ($mean = 0.97$) for 18 of 20 products. The two exceptions were the Belkin Switch and Sonos Play. Using pattern representations, all 20 products again achieved an MFAF greater than 0.9 while maintaining a relatively small attack surface. Pattern representations also had clear advantages over hostname representations at higher thresholds for a few products. A higher threshold equates to a stricter allowlist, which makes the allowlist more robust to noisy or poisoned data. When the threshold is set to 5, only 10 products have an MFAF of at least 0.9 when using hostname representations, but 15 products have an MFAF of at least 0.9 when using pattern representations. Emphasizing these trends, Figure 6.3 shows the six products whose MFAF increases more than 0.1 by switching from hostnames to patterns. Appendix 7.9 shows similar trends hold when analyzing a larger dataset of 52 products.

6.4.4 Adding Ports to Host Representations

In most cases, adding ports to the host representation minimally impacts the MFAF despite further reducing the attack surface. For example, for hostname, pattern, and domain host representations with $threshold = 1$, 19 out of 20 products' MFAF drop less than 2%. For the

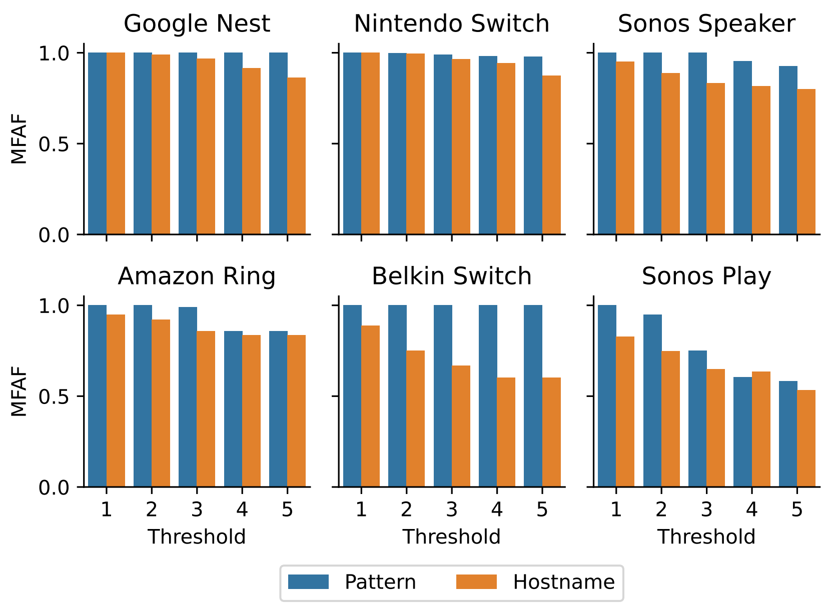


Figure 6.3: MFAF for pattern- and hostname-based allowlists.

remaining product, Amazon Ring, the MFAF still drops less than 5%. When $threshold = 1$, the pattern-based allowlist for the Amazon Ring shows a decrease of 4.4% in MFAF after adding ports. With domain-based and hostname-based allowlist, the drops are 4.2% and 2.2%, respectively. The decrease is consistent with the previous observation (Section 6.3) that Amazon Rings communicates with `amazonaws.com` on seemingly random remote ports. The decrease is smaller for hostname-based allowlists because the number of load balancing servers is large for `amazonaws.com`, resulting in fewer ports observed on each server.

6.4.5 Sample Size Required

Using more devices of a given product to generate an allowlist makes it increasingly likely the allowlist will capture the long tail of endpoints with which a product communicates. Thus, we calculated the MFAF for various sample sizes. Specifically, we altered the size of the training sample with a step size of five, observing how the MFAF changed. For stability, we performed five-fold cross validation per product and repeated this process five times, taking the mean.

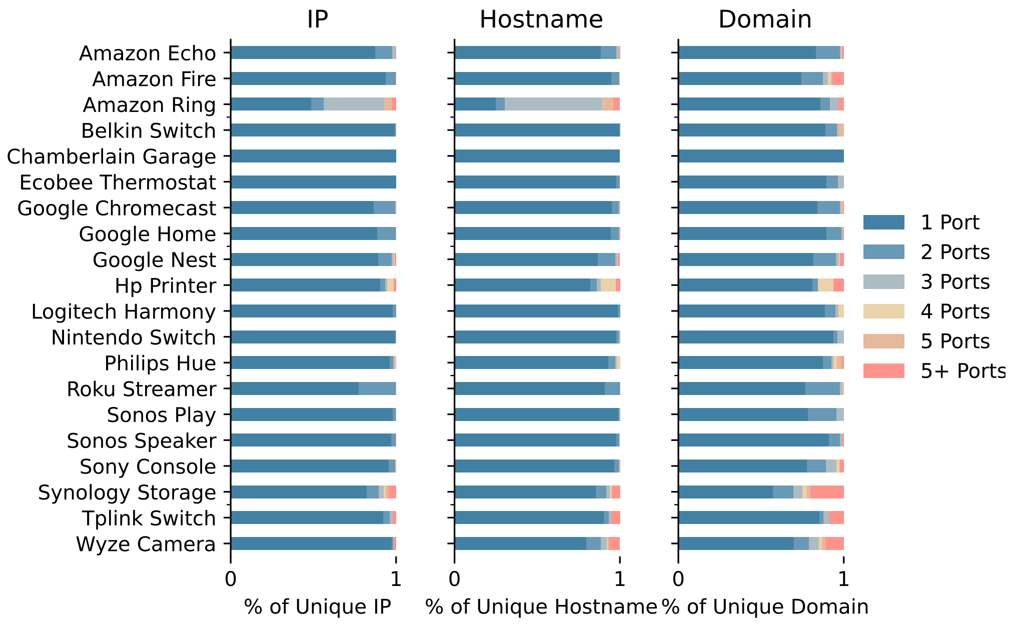


Figure 6.4: The number of ports used by remote hosts with IP address, hostname, and domain host representations. Few remote hosts used multiple ports for incoming connections.

Figure 6.5 shows the results. In general, hostname representations necessitate a larger sample than pattern representations. For hostname-based allowlists, only 15 of the 20 products ever achieved an MFAF of at least 0.95, and it took 28 devices on average to do so. In contrast, pattern-based allowlists enabled all products to achieve an MFAF of at least 0.95, requiring only 12 devices on average. These numbers, however, varied significantly across products, particularly for hostname representations. For example, 5 Ecobee Thermostats were sufficient to form a hostname-based allowlist, but 185 Sonos Speakers were needed for the same MFAF.

Pattern representations were critical when the sample size was small due to a lack of data. For example, the dataset contained only a few HP printers, and HP uses load balancing for XMPP (e.g., with `xmpp001.hpeprint.com` and `xmpp002.hpeprint.com`). With at least two related hostnames observed, pattern representations can thus potentially compensate for a smaller sample.

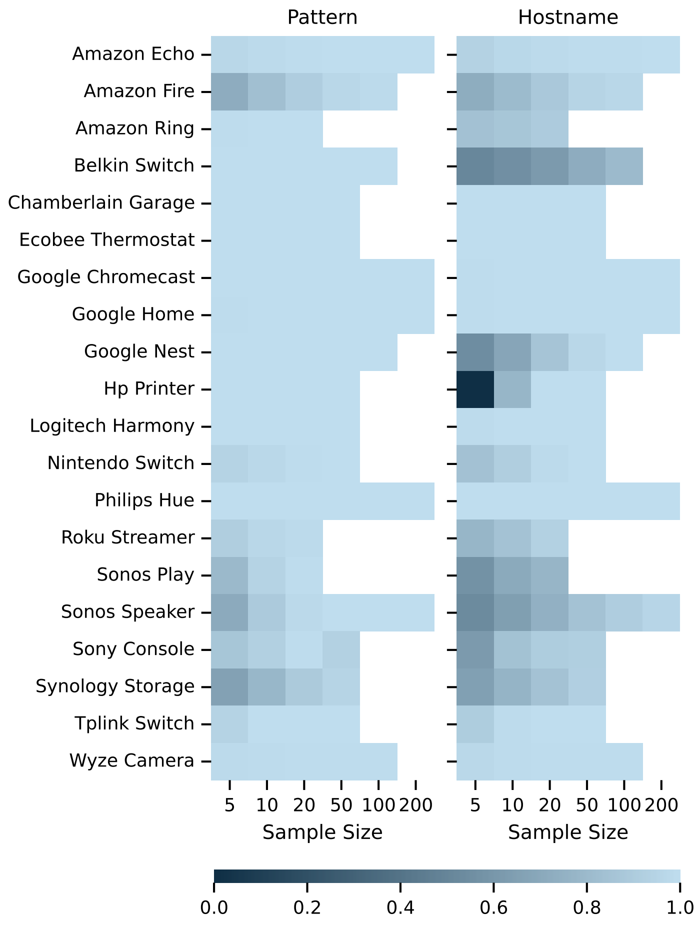


Figure 6.5: The MFAF increases with the training sample size ($threshold = 1$). Blank squares indicate insufficient data.

6.4.6 Setting the Threshold

The threshold, specifying on how many different devices an endpoint must be observed for it to be added to the allowlist, represents a direct trade-off between security and generalizability. Increasing the threshold typically admits fewer endpoints, but both shrinks the attack surface and increases robustness to a few devices being mislabeled or compromised. 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 without losing much generalizability.

To understand the relationship between thresholds and the MFAF, we sampled 50 devices

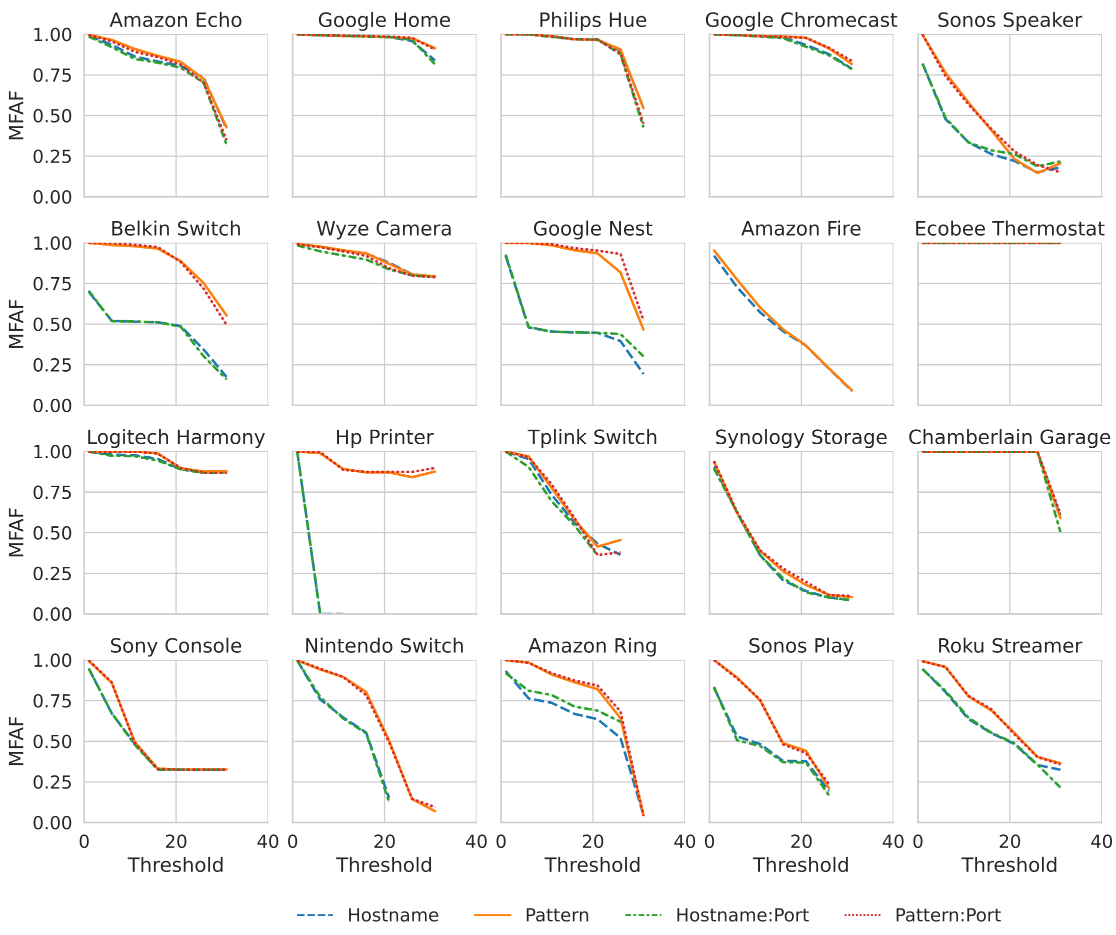


Figure 6.6: Although the MFAF generally decreases with increasing thresholds, many products experience plateaus where the thresholds increase, but the corresponding MFAF changes little. Lines end when the allowlist is blank above that threshold.

from each product, calculating the MFAF 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 6.6, some products effectively had plateaus in which increasing the threshold minimally impacted the MFAF. In one of the more extreme examples, the MFAF for the 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 contacted by few devices. Only two endpoints were contacted by the majority of those devices: `connect1.myqdevice.com` and `connect.myqdevice.com`. In contrast, products like the Amazon Fire and Roku Streamer that rely on streaming user-specified content rarely exhibit this plateau.

6.4.7 Accounting For Regionalization

As previously mentioned, IoT Inspector primarily consists of devices from North and South America. To better understand how geography and regionalization impact allowlists' MFAF, we created a series of regression models that gauge how region impacts MFAF. Because IoT Inspector does not contain location information, we use 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 MFAF 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 7.5- 7.7 in Appendix 7.7 contains the regression tables. All coefficients of the train region, test region, and their interactions were significant. At a high level, applying an allowlist generated with training data from one region to testing data from a different region negatively impacts the MFAF. Furthermore, regions differ in the MFAF of the allowlists generated even when keeping the sample size constant. For example, applying allowlists generated from devices in Region A to devices in Region C caused a drop of around 0.1 in MFAF. Therefore, if the data is geographically diverse, care should be paid to matching the training data and test data; home IoT devices' network behaviors vary partially by region.

6.4.8 Transferring Allowlists By Vendor

To this point, we created allowlists for a product from the network behaviors of devices of that same product. While this is the most typical setting, newly introduced or rare devices

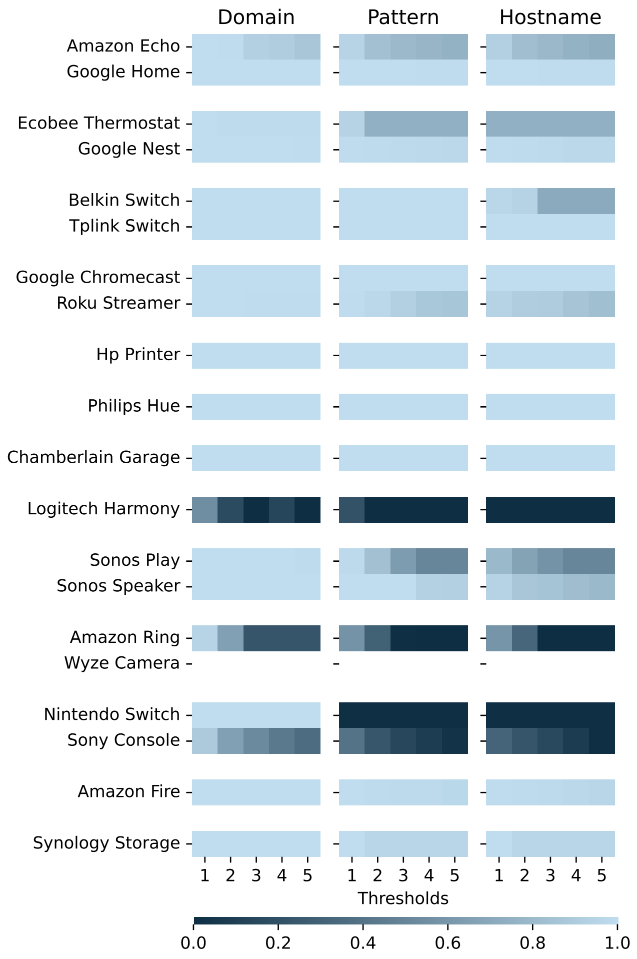


Figure 6.7: The MFAF of generating an allowlist for a product from only *other products* from the same vendor.

may lack sufficient training data, raising the question of whether it is possible to transfer an allowlist generated for one product to a *different product made by the same vendor*. For example, could an allowlist generated for Google Home be successfully applied to Google Nest devices?

At a high level, we found that domain-based allowlists can sometimes be transferred to other products from the same vendor. Exceptions include IoT cameras and some vendors who make a wide range of products. Furthermore, pattern-based and hostname-based allowlists transferred somewhat less well across products. Figure 6.7 was created by using the product listed as the testing data and all other products from the same vendor as the training data to

Table 6.1: How host representations impact the attack surface, specifically the number of hostnames in IoT Inspector allowed if one adds the given representation to the allowlist.

Example Hostnames, Patterns, and Domains	**# Hosts**
Hostname: **arlostatic-z1.s3.amazonaws.com**	1
• Pattern: *arlostatic-z[0-9]+\.s3\.amazonaws\.com*	3
• Domain: *amazonaws.com*	3859
Hostname: **ring-untranscoded-videos.s3.amazonaws.com**	1
• Pattern: *ring-(—un)transcoded-videos(-lq—)\.s3\.amazonaws\.com*	2
• Domain: *amazonaws.com*	3859
Hostname: **a191avoddashs3ww-a.akamaihd.net**	1
• Pattern: *a[0-9]+avoddashs[0-9]+(ww—us)-a\.akamaihd\.net*	50
• Domain: *akamaihd.net*	96

compute the MFAF. Comparing Figure 6.2 and Figure 6.7 shows that 15 products' allowlists did transfer across products made by the same vendor, typically because those products share API endpoints.

Cameras were a key exception, as shown in Figure 7.2 in Appendix 7.9. Grouped by device type, this figure compares how MFAF differs when a product's allowlist is applied to the product itself, as well as other products made by the same vendor. We found that allowlists created for cameras and speakers struggle the most when applied to other products from the same vendor. One explanation is that the infrastructure and endpoints required by a camera or speaker are very different from those required by a switch or light.

One example of vendor transferability failing is when a product line is purchased by another company or when a company relies on a third party for a specific feature. These cases often result in products with unique-for-that-vendor behaviors. For example, most of Logitech's Harmony remote controllers' requests are sent to `pubnub.com`, a third-party company that provides real-time communication and is argued to be critical to Harmony's away-from-home capability [232]. Amazon Ring is another example. The Ring line was purchased by Amazon in 2018 [233], but kept its own endpoints.

6.4.9 Attack Surface and Evasion

As mentioned in Section 6.1, we assume attackers know the contents of the allowlist. Depending on which host representation and threshold the user chooses, as well as how a vendor deploys their public cloud services, an attacker may be able to evade the allowlist. While domains are more generalizable than hostnames or patterns, they can easily fall short in security, particularly due to home IoT devices' reliance on cloud providers.

To provide intuition for how the host representation impacts the attack surface, Table 6.1 contrasts hostname, pattern, and domain representations for three example endpoints in the dataset. 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. Whereas the pattern abstracted from these hostnames matches three different hostnames in the dataset, extending the pattern to the domain `amazonaws.com` matches 3,859. It is then important to consider whether an attacker can register or otherwise obtain an illegitimate endpoint that matches either the pattern (less likely) or domain (far more likely). In this example, relying on domains, rather than patterns, widens the attack surface and brings all of `amazonaws.com` into the attack surface. The attacker can easily host malicious content somewhere on AWS or direct a DDoS attack toward victims who themselves use AWS. Using a pattern representation often provides a far narrower attack surface than using a domain while still supporting generalizability. That said, hostname patterns can have complex security implications. For *arlostatic-z[0-9]+\.s3\.amazonaws\.com*, 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 the regular expression. Without registration restrictions from the cloud service provider or extra diligence from the vendors, registering such a hostname may be possible for pattern and domain representations. Unfortunately, as discussed above, hostname-based allowlists are the most likely to inadvertently block traffic from unseen, but related, endpoints.

6.5 Enforcing Allowlists in the Lab

To test whether the allowlists generated in various circumstances impaired the functionality of devices, we deployed allowlists with various thresholds and host representations on nine devices in our lab. We intentionally exercised as many functionalities as possible per device. We found that many of these functionalities seemed to work as intended, though we observed and diagnosed a few different failure modes. While we worried that rare behaviors (e.g., factory resets) unlikely to have been captured in IoT Inspector might prove problematic, this worry did not manifest in practice. Finally, the success of deploying these allowlists in our lab despite a two-year gap since IoT Inspector was collected highlights the stability of the allowlists generated.

6.5.1 Implementation

To enforce our generated allowlists on real devices, we implemented our own firewall through `scapy`, a Python library that manipulates network packets, and `NetfilterQueue`, a Python library that intercepts network packets using `iptables`. We have open-sourced our firewall implementation in our Github repository, anonymized for review [228]. The firewall runs on a Jetson Nano, which served as a NAT gateway. In that way, all traffic generated from the home IoT devices can be intercepted and manipulated by our firewall. The workflow of our implementation is as follows.

Device Configuration When a new device is added to the network, the user must create a configuration file for the device. The firewall can thus recognize the device and intercept its traffic accordingly. The configuration file needs the following information: (i) a unique *device name*; (ii) the device's *MAC address* so the firewall can recognize the device as `NetfilterQueue` intercept traffics via `iptables`, which supports MAC addresses; and (iii) the *product name* to apply the appropriate allowlist. Optionally, the device's *static IP* *address* can be used in place of the MAC address.

Allowlist Configuration As discussed in Section 6.4, several parameters can be used to customize the allowlist. The *threshold* parameter adjusts the restrictiveness of an allowlist. One can also specify which *host representation* to use. We support hostname, domain, pattern, and IP address representations. Optionally, one could also specify if they want their allowlist to be region-specific, though by default all data from all regions will be used. Appendix 7.8 contains an example allowlist. We generate the relevant allowlists prior to the deployment in the lab. In practice, we expect the user will download their chosen allowlist and deploy it on a router.

Allowlist Enforcement `NetfilterQueue` intercepts traffic using `iptables`, which means it only naturally supports IP addresses and ports. However, as described in Section 6.4, hostnames and patterns are our preferred representations. In some cases, domain-based filtering is also necessary. Therefore, we take extra steps to enable our firewall to handle these representations as well.

The firewall uses the router's default DNS resolver to resolve any hostname-based rules. All hostname rules will be resolved to IP addresses and added to the IP-based allowlists for enforcement. If the queried hostname in the DNS query is on the allowlist, then the firewall will forward the query to the DNS resolver. Otherwise, the firewall will drop the DNS query. Once a DNS response is received, the firewall will record the included IP addresses before forwarding it to the device. Domain-based rules are handled similarly. The only difference is that the DNS queries are checked against domain-based allowlists instead of those based on hostnames.

Neither `iptables` nor DNS supports patterns, which means the firewall needs to interpret patterns by itself. If patterns are used, the queried hostname in DNS traffic will be matched on the router against the patterns allowed. Since the allowlist can be quite long, the firewall will first find candidates most similar to the queried hostname and check whether there is a match. If there is a match, the DNS query will be forwarded; otherwise, it will be dropped.

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. Once the firewall sees the DNS query and response, it will correct its behaviors and resume normal operation.

6.5.2 Experimental Setup and Protocol

For our in-lab experiment, we selected the nine products that appeared most frequently in the IoT Inspector dataset. For highly similar products (e.g., Amazon Echo and Amazon Echo Dot), we tested only one. All devices were connected to the aforementioned Jetson Nano over WiFi. The Jetson Nano, running Ubuntu 20.04, was used to intercept devices' network traffic and enforce the allowlist.

Before running the tests, two researchers collaboratively determined each product's main functionalities and purposes using the product's manuals and companion app. See Appendix 7.6 for a complete list of functionalities. We excluded functionalities that do not center on the device's main purpose, such as changing the device name. We also excluded functionalities that do not require Internet connectivity, such as casting a webpage to a Google Chromecast.

We first explored the expected behavior of each product without allowlists, including rare operations (e.g., factory reset). The intention was to establish a baseline by documenting the behavior of each product's functionalities without allowlists. We then applied our allowlists to each product, proceeding from the most permissive to the most restrictive by increasing the threshold, as defined in Section 6.4. Once an allowlist was deployed, we tried to exercise each of the product's documented functionalities. If any functionality failed, we repeated the experiment to confirm. The process ended either when all functionalities failed or the most restrictive allowlist was tested. Whenever possible, we tested each device from an external network (i.e., interacting with the companion app outside the lab's network) to maximize the product's Internet exposure. Only when devices cannot 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 since most devices we tested are compatible with Alexa.

Our approach differs from that of Mandalari et al. [69], who also tested IoT devices in a lab. They studied blocklists for improving privacy; we evaluate allowlists for improving security. Further, Mandalari et al. obtained IoT destinations based on in-lab measurement, whereas our method used IoT Inspector as training data.

6.5.3 Key Results

Of the 52 functionalities tested from the 9 devices, 50 of them worked with at least one allowlist generated from IoT Inspector. 32 functionalities worked with a hostname-based allowlist, and 43 functionalities worked with pattern allowlists. In contrast, 50 of the 52 functionalities worked with domain-based allowlists. Table 6.2 details our results, including the maximum threshold at which the functionality worked and the size of the allowlist generated.

Dedicated Hostnames In some cases, compact allowlists were sufficient; 17 functionalities (33%) still worked when deploying an allowlist with fewer than 10 entries. These functionalities were typically supported by dedicated hostnames, making it possible to apply a hostname-based allowlist. For instance, Google Home only needs `www.google.com` for all functionalities other than media (e.g., playing music). The actual execution, such as controlling other devices or querying external information (e.g., weather), is handled through

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	
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
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 Chromecast (N=288)	Watch Live TV	× CDNs		230	2	47	5
	Factory reset	13	115	5	201	2	201
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

Table 6.2: The highest tested thresholds (i.e., the strictest allowlist) and the number of allowed hostnames/patterns/domains that still keep the devices functioning in our in-lab experiment. × [Failure Reason] means none of our generated allowlists keep the functionality running; the "failure reason" explains why. To force the tested devices to use the Internet instead of the local network for functionality, 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). ***Wyze camera* is marked with a * as the device only works with some additional manually allowed IP addresses. The IP addresses are the main contributors to the Wyze Camera's traffic and can be easily observed, but no associated DNS lookup was observed.**

Device	Functionality	Hostname		Hostname Pattern		Domain	
		# of Allowed Hostnames	Threshold	# of Allowed Patterns	Threshold	# of Allowed Domains	Threshold
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
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
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

Table 6.3: (Continued) The highest tested thresholds (i.e., the strictest allowlist) and the number of allowed hostnames/patterns/domains that still keep the devices functioning in our in-lab experiment. × [Failure Reason] means none of our generated allowlists keep the functionality running; the "failure reason" explains why. To force the tested devices to use the Internet instead of the local network for functionality, 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).

server-to-server communication.

Streaming Media For streaming audio or video, hostname-based allowlists struggled. Of the 11 streaming functionalities tested, 7 did not work with any hostname-based allowlists. Among these 7 functionalities, 4 started working after we switched to a pattern-based allowlist, and two others only worked with a domain-based allowlist. The final functionality, Sonos Radio, was a newly introduced feature. The endpoint it contacts, `sonos.radio`, was not present in the two-year-old IoT Inspector dataset.

Ports An allowlist that includes ports may necessitate a lower threshold for some devices because the number of devices that contact a specific port on a host is lower than the number of devices that contact the host overall. We confirm this through a post-hoc analysis. For example, according to the IoT Inspector dataset, `connectivitycheck.gstatic.com`, an essential endpoint for Google Home to function, operates both on port 80 and port 443. The dataset shows that more devices use port 443, whereas our Google Home uses port 80 instead. As a result, if ports are included for a Google Home allowlist, a lower-threshold allowlist or ad-hoc workaround (e.g., always allowing port 80) is required.

Allowlist Stability Despite the aforementioned Sonos example, our results overall suggest that allowlists can be quite stable, at least for core functionalities. Among the nine products tested, only three (Amazon Fire Stick, Belkin Wemo switch, and Sonos One) had functionalities seemingly impacted by the age of the IoT Inspector dataset. While the endpoint for the Belkin Wemo had changed in the intervening two years, severely impacting functionality when allowlists were deployed, the other two devices were affected in only minor ways. We also noticed some hosts change their ports (e.g., `drift.com`, used by Google Home, changed from port 443 to 9999). However, none of these endpoints appeared to be essential to functionality. Most vendors still used the same endpoints/APIs, making allowlists fairly stable despite multiple firmware updates.

6.5.4 Reasons For Failures

To inform the design of future allowlists, we investigated and attributed the root causes of particular functionalities failing.

Content Delivery Networks (CDNs) Among the 20 functionalities hostname-based allowlists break, the use of CDNs was directly responsible for six, with another four being affected indirectly (e.g., pausing music is not relevant if the music cannot be loaded in the first place). CDN hostnames commonly included seemingly random numbers or letters. As a result, allowlists based on exact hostname matches frequently blocked legitimate communication.

Our lab experiments verified that pattern-based allowlists often mitigate this failure. For example, when we attempted to play a playlist, Sonos One contacted `guc3-accesspoint-a-vr15.ap.spotify.com`, which was blocked by hostname-based allowlists, but allowed by pattern-based ones. We also observed cases where hostnames were too random to be clustered by DBSCAN, such as hostnames from CloudFront (e.g., `d924eisv4ltzs.cloudfront.net`). If the hostname is not included in the IoT Inspector dataset, then it is necessary to have a domain-based rule instead.

Load Balancing Even with a large dataset like IoT Inspector, not all load-balancing endpoints will be observed. The Amazon Echo Dot in the lab experiments contacted `avs-alexa-14-na.amazon.com` for Alexa services, which was not observed in the dataset. However, as many other Alexa AVS endpoints were observed in the dataset, a successful pattern was extracted, solving the problem.

Dependencies Some functionalities were impaired due to blocking another functionality. For example, as mentioned previously, hostname-based allowlists often impaired music streaming, including for Sonos One. All other functionalities, such as changing the volume, were thus no longer accessible. Connectivity checking was another common, and important, dependency. Many devices, such as the Google Home and Belkin Wemo Switch, perform connectivity checking. If it fails, then the device refuses to take any further commands until it believes it is again online. 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.

API Changes Rarely, vendors will change their API and infrastructure, invalidating allowlists generated from past data. This happened with the Belkin Wemo Switch. 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 [26, 69]. However, Belkin recently added a new hostname, `deviceapis.xwemo.com`. Failing to allow this new, previously unseen API endpoint put the device offline. Ironically, the original API endpoint was still live, and the switch still contacted it from time to time.

New Services In addition to changes to API endpoints, the vendor may also add new features. In our case, Sonos Radio was a new feature launched in 2020 [234]. In the lab, Sonos radio required a new, previously unobserved domain (`sonos.radio`) for communication.

Third-Party Services We anticipated that integration with third-party services chosen by users could be another source of failures, though we did not observe this to be the case in our experiments. Popular integrations with YouTube, Netflix, or Spotify had been observed in IoT Inspector. However, users who prefer less popular services may encounter problems.

6.5.5 Limitations

Because our firewall implementation is simply a proof-of-concept to evaluate allowlists' real-world impact on device functionality, we did not design the implementation for high performance nor conduct a performance evaluation. Although we did not experience any human-noticeable delay during the experiments, we note that our prototype intercepts traffic in user space instead of kernel space, likely introducing latency. To enable allowlists that support hostnames, domains, and hostname patterns, it is inevitable to intercept DNS traffic for matching. However, the IP addresses mapped to these representations can be easily written as *iptables* rules, which can speed up the process with proper caching.

6.6 Discussion

Home IoT devices have a significant attack surface due to the prevalence of unpatched security vulnerabilities. Although relying on vendors to secure devices through regular security updates offers one level of defense, such an approach is incomplete. Not all vendors regularly apply updates. In some cases, IoT devices remain connected well past when vendors support them. This state of affairs necessitates network-level defenses. Unfortunately, determining precisely which traffic flows should be allowed from any particular device is challenging due to the surprising heterogeneity of behavior across home IoT devices—even devices of exactly the same product. Because of this heterogeneity, generating generalizable allowlists that minimize the attack surface is the key challenge.

This chapter has explored the design space for home IoT network allowlists, evaluating how factors like the representation of a network endpoint (i.e., hostname, domain, pattern), geographic location, and amount of observed traffic from the device can contribute to the construction of an allowlist that generalizes across devices of the same type or vendor. Our evaluation is promising, suggesting that generalizable allowlists based on hostname patterns can be constructed for many products based on a training sample of only one or two dozen devices. We also discovered that localizing the training data by region can be helpful since devices exhibit geographically specific behaviors. Our evaluation of the allowlists we generated on real devices in a lab study—two years after the training set was collected—demonstrated that such allowlists generally work, reducing the attack surface without significantly impairing functionality. IoT devices that rely on cloud services often contact hostnames that include seemingly random numbers or patterns, making generalizability difficult and opening a security hole as domain-based representations can be quite broad. Vendors employing more careful strategies for generating and assigning these hostnames in the future could further manage this tradeoff.

REFERENCES

[1] S. Mattu and K. Hill. The house that spied on me, 2018. `https://gizmodo.com/the-house-that-spied-on-me-1822429852`.

[2] Eric Zeng, Shrirang Mare, and Franziska Roesner. End User Security and Privacy Concerns with Smart Homes. In *Proc. SOUPS*, 2017.

[3] Florian Schaub, Rebecca Balebako, Adam L. Durity, and Lorrie Faith Cranor. A Design Space for Effective Privacy Notices. In *Proc. SOUPS*, 2015.

[4] 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 *Proc. IMC*, 2019.

[5] Nan Zhang, Soteris Demetriou, Xianghang Mi, Wenrui Diao, Kan Yuan, Peiyuan Zong, Feng Qian, XiaoFeng Wang, Kai Chen, Yuan Tian, Carl A. Gunter, Kehuan Zhang, Patrick Tague, and Yue-Hsun Lin. Understanding IoT Security Through the Data Crystal Ball: Where We Are Now and Where We Are Going to Be. *CoRR*, abs/1703.09809, 2017.

[6] 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 *Proc. HotNets*, 2015.

[7] Omar Alrawi, Chaz Lever, Manos Antonakakis, and Fabian Monrose. Sok: Security evaluation of home-based iot deployments. In *Proc. IEEE SP*, 2019.

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

[9] Stuart Schechter. The User IS the Enemy, and (S)he Keeps Reaching for that Bright Shiny Power Button! In *Proc. HUPS*, 2013.

[10] Blase Ur, Jaeyeon Jung, and Stuart Schechter. Intruders Versus Intrusiveness: Teens' and Parents' Perspectives on Home-entryway Surveillance. In *Proc. UbiComp*, 2014.

[11] Eric Zeng and Franziska Roesner. Understanding and improving security and privacy in multi-user smart homes: A design exploration and in-home user study. In *Proc. USENIX Security Symposium*, 2019.

[12] Vassilios Lekakis, Yunus Basagalar, and Pete Keleher. Don't Trust Your Roommate or Access Control and Replication Protocols in "Home" Environments. In *Proc. HotStorage*, 2012.

[13] A.J. Bernheim Brush, Jaeyeon Jung, Ratul Mahajan, and Frank Martinez. Digital Neighborhood Watch: Investigating the Sharing of Camera Data Amongst Neighbors. In *Proc. CSCW*, 2013.

[14] Christine Geeng and Franziska Roesner. Who's in control? Interactions in multi-user smart homes. In *Proc. CHI*, 2019.

[15] Aaron Tilley. How A Few Words To Apple's Siri Unlocked A Man's Front Door, September 2016. `https://www.forbes.com/sites/aarontilley/2016/09/21/apple-homekit-siri-security`, as of June 27, 2022.

[16] John Patrick Pullen. Amazon Echo Owners Were Pranked by South Park and Their Alexas Will Make Them Laugh for Weeks, September 2017. `http://fortune.com/2017/09/14/watch-south-park-alexa-echo/`, as of June 27, 2022.

[17] Venessa Wong. Burger King's New Ad Will Hijack Your Google Home, April 2017. `https://www.cnbc.com/2017/04/12/burger-kings-new-ad-will-hijack-your-google-home.html`, as of June 27, 2022.

[18] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. Robust physical-world attacks on deep learning visual classification. In *Proc. CVPR*, 2018.

[19] Guoming Zhang, Chen Yan, Xiaoyu Ji, Tianchen Zhang, Taimin Zhang, and Wenyuan Xu. DolphinAttack: Inaudible voice commands. In *Proc. CCS*, 2017.

[20] Nicholas Carlini and David A. Wagner. Audio adversarial examples: Targeted attacks on speech-to-text. In *Proc. DLS*, 2018.

[21] Hyunwoo Yu, Jaemin Lim, Kiyeon Kim, and Suk-Bok Lee. Pinto: Enabling video privacy for commodity IoT cameras. In *Proc. CCS*, 2018.

[22] Yuxin Chen, Huiying Li, Shan-Yuan Teng, Steven Nagels, Zhijing Li, Pedro Lopes, Ben Y. Zhao, and Haitao Zheng. Wearable microphone jamming. In *Proc. CHI*, 2020.

[23] Milijana Surbatovich, Jassim Aljuraidan, Lujo Bauer, Anupam Das, and Limin Jia. Some Recipes Can Do More Than Spoil Your Appetite: Analyzing the Security and Privacy Risks of IFTTT Recipes. In *Proc. WWW*, 2017.

[24] Z. Berkay Celik, Gang Tan, and Patrick D. McDaniel. IoTGuard: Dynamic enforcement of security and safety policy in commodity IoT. In *Proc. NDSS*, 2019.

[25] Weijia He, Valerie Zhao, Olivia Morkved, Sabeeka Siddiqui, Earlence Fernandes, Josiah Hester, and Blase Ur. SoK: Context sensing for access control in the adversarial home iot. In *Proc. EuroS&P*, 2021.

[26] Danny Yuxing Huang, Noah J. Apthorpe, Frank Li, Gunes Acar, and Nick Feamster. Iot inspector: Crowdsourcing labeled network traffic from smart home devices at scale. *IMWUT*, 4(2), 2020.

[27] Earlence Fernandes, Jaeyeon Jung, and Atul Prakash. Security Analysis of Emerging Smart Home Applications. In *Proc. IEEE SP*, 2016.

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

[29] Qi Wang, Wajih Ul Hassan, Adam Bates, and Carl Gunter. Fear and Logging in the Internet of Things. In *Proc. NDSS*, 2018.

[30] Behrang Fouladi and Sahand Ghanoun. Honey, I'm Home!!, Hacking ZWave Home Automation Systems, July 2013. Black Hat USA.

[31] 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 *Proc. USENIX Security Symposium*, 2017.

[32] Yunhan Jack Jia, Qi Alfred Chen, Shiqi Wang, Amir Rahmati, Earlence Fernandes, Z. Morley Mao, and Atul Prakash. ContexloT: Towards Providing Contextual Integrity to Appified IoT Platforms. In *Proc. NDSS*, 2017.

[33] Yuan Tian, Nan Zhang, Yueh-Hsun Lin, Xiaofeng Wang, Blase Ur, Xianzheng Guo, and Patrick Tague. SmartAuth: User-Centered Authorization for the Internet of Things. In *Proc. USENIX Security Symposium*, 2017.

[34] Michelle L. Mazurek, J. P. Arsenault, Joanna Bresee, Nitin Gupta, Iulia Ion, Christina Johns, Daniel Lee, Yuan Liang, Jenny Olsen, Brandon Salmon, Richard Shay, Kami Vaniea, Lujo Bauer, Lorrie Faith Cranor, Gregory R. Ganger, and Michael K. Reiter. Access Control for Home Data Sharing: Attitudes, Needs and Practices. In *Proc. CHI*, 2010.

[35] Tiffany Hyun-Jin Kim, Lujo Bauer, James Newsome, Adrian Perrig, and Jesse Walker. Access Right Assignment Mechanisms for Secure Home Networks. *Journal of Communications and Networks*, 13(2):175–186, 2011.

[36] Tiffany Hyun-Jin Kim, Lujo Bauer, James Newsome, Adrian Perrig, and Jesse Walker. Challenges in Access Right Assignment for Secure Home Networks. In *Proc. HotSec*, 2010.

[37] Lujo Bauer, Lorrie Faith Cranor, Robert W. Reeder, Michael K. Reiter, and Kami Vaniea. Real Life Challenges in Access-control Management. In *Proc. CHI*, 2009.

[38] Elizabeth Stobert and Robert Biddle. Authentication in the Home. In *Proc. HUPS*, 2013.

[39] Blase Ur, Jaeyeon Jung, and Stuart Schechter. The Current State of Access Control for Smart Devices in Homes. In *Proc. HUPS*, 2013.

[40] Pardis Emami Naeini, Sruti Bhagavatula, Hana Habib, Martin Degeling, Lujo Bauer, Lorrie Faith Cranor, and Norman Sadeh. Privacy Expectations and Preferences in an IoT World. In *Proc. SOUPS*, 2017.

[41] Matthew Johnson and Frank Stajano. Usability of Security Management: Defining the Permissions of Guests. In *Proc. SPW*, 2006.

[42] Tamara Denning, Tadayoshi Kohno, and Henry M. Levy. Computer Security and the Modern Home. *CACM*, 56(1):94–103, 2013.

[43] Roei Schuster, Vitaly Shmatikov, and Eran Tromer. Situational access control in the Internet of Things. In *Proc. CCS*, 2018.

[44] Simon Birnbach, Simon Eberz, and Ivan Martinovic. Peeves: Physical event verification in smart homes. In *Proc. CCS*, 2019.

[45] Eun Kyoung Choe, Sunny Consolvo, Jaeyeon Jung, Beverly Harrison, Shwetak N. Patel, and Julie A. Kientz. Investigating receptiveness to sensing and inference in the home using sensor proxies. In *Proc. UbiComp*, 2012.

[46] Jalal Al-Muhtadi, Anand Ranganathan, Roy Campbell, and M. Dennis Mickunas. A Flexible, Privacy-Preserving Authentication Framework for Ubiquitous Computing Environments. In *Proc. ICDCS*, 2002.

[47] Jing Liu, Yang Xiao, and C.L. Philip Chen. Authentication and Access Control in the Internet of Things. In *Proc. ICDCS*, 2012.

[48] Huan Feng, Kassem Fawaz, and Kang G. Shin. Continuous Authentication for Voice Assistants. In *Proc. MobiCom*, 2017.

[49] Adrienne Porter Felt, Elizabeth Ha, Serge Egelman, Ariel Haney, Erika Chin, and David Wagner. Android Permissions: User Attention, Comprehension, and Behavior. In *Proc. SOUPS*, 2012.

[50] Primal Wijesekera, Arjun Baokar, Ashkan Hosseini, Serge Egelman, David Wagner, and Konstantin Beznosov. Android Permissions Remystified: A Field Study on Contextual Integrity. In *Proc. USENIX Security Symposium*, 2015.

[51] William Enck, Machigar Ongtang, and Patrick McDaniel. Understanding Android Security. *IEEE Security & Privacy*, 7(1):50–57, 2009.

[52] Adrienne Porter Felt, Serge Egelman, and David Wagner. I've Got 99 Problems, but Vibration Ain't One: A Survey of Smartphone Users' Concerns. In *Proc. SPSM*, 2012.

[53] Google. Android Supporting Multiple Users, June 2017. `https://source.android.com/devices/tech/admin/multi-user`, as of June 27, 2022.

[54] Josephine Lau, Benjamin Zimmerman, and Florian Schaub. Alexa, are you listening?: Privacy perceptions, concerns and privacy-seeking behaviors with smart speakers. *PACM HCI*, 2(CSCW), 2018.

[55] Nathan Malkin, Julia Bernd, Maritza Johnson, and Serge Egelman. "What can't data be used for?" Privacy expectations about smart TVs in the US. In *Proc. EuroUSEC*, 2018.

[56] Noah Apthorpe, Pardis Emami-Naeini, Arunesh Mathur, Marshini Chetty, and Nick Feamster. You, me, and IoT: How internet-connected consumer devices affect interpersonal relationships. arXiv:2001.10608, 2020.

[57] Yaxing Yao, Justin Reed Basdeo, Smirity Kaushik, and Yang Wang. Defending my castle: A co-design study of privacy mechanisms for smart homes. In *Proc. CHI*, 2019.

[58] Emilee Rader and Janine Slaker. The importance of visibility for folk theories of sensor data. In *Proc. SOUPS*, 2017.

[59] Timo Jakobi, Sameer Patil, Dave Randall, Gunnar Stevens, and Volker Wulf. It is about what they could do with the data: A user perspective on privacy in smart metering. *TOCHI*, 26(1), 2019.

[60] Timo Jakobi, Gunnar Stevens, Nico Castelli, Corinna Ogonowski, Florian Schaub, Nils Vindice, Dave W. Randall, Peter Tolmie, and Volker Wulf. Evolving needs in IoT control and accountability: A longitudinal study on smart home intelligibility. *IMWUT*, 2(4), 2018.

[61] Nata M. Barbosa, Joon S. Park, Yaxing Yao, and Yang Wang. "What if?" Predicting individual users' smart home privacy preferences and their changes. *PoPETS*, 2019(4):211–231, 2019.

[62] Chuhan Gao, Kassem Fawaz, Sanjib Sur, and Suman Banerjee. Privacy protection for audio sensing against multi-microphone adversaries. *PoPETs*, 2019(2):146–165, 2019.

[63] Vasilios Mavroudis, Shuang Hao, Yanick Fratantonio, Federico Maggi, Christopher Kruegel, and Giovanni Vigna. On the privacy and security of the ultrasound ecosystem. *PoPETs*, 2017(2):95–112, 2017.

[64] Mariella Dimiccoli, Juan Marín, and Edison Thomaz. Mitigating bystander privacy concerns in egocentric activity recognition with deep learning and intentional image degradation. *IMWUT*, 1(4):132:1–132:18, 2017.

[65] Rakibul Hasan, David Crandall, Mario Fritz, and Apu Kapadia. Automatically detecting bystanders in photos to reduce privacy risks. In *Proc. IEEE S&P*, 2020.

[66] Earlence Fernandes, Amir Rahmati, Kevin Eykholt, and Atul Prakash. Internet of Things security research: A rehash of old ideas or new intellectual challenges? *IEEE Security & Privacy*, 15(4):79–84, 2017.

[67] Chen Yan, Hocheol Shin, Connor Bolton, Wenyuan Xu, Yongdae Kim, and Kevin Fu. SoK: A minimalist approach to formalizing analog sensor security. In *Proc. IEEE S&P*, 2020.

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

[69] 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.

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

[71] 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.

[72] 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.

[73] 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.

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

[75] Corentin Thomasset and David Barrera. Sereniot: Distributed network security policy management and enforcement for smart homes. In *Proc. ACSAC*, pages 542–555. ACM, 2020.

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

[77] Vern Paxson. Bro: A system for detecting network intruders in real-time. *Computer networks*, 31(23-24):2435–2463, 1999.

[78] 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.

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

[80] 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.

[81] 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.

[82] 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.

[83] 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.

[84] 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.

[85] 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.

[86] 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.

[87] Bitdefender. Bitdefender box, 2022. `https://mastercard.bitdefender.com/box/`.

[88] NortonLifeLock Inc. Norton core router, 2018. `https://us.norton.com/core`.

[89] 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.

[90] Aman Singh, Shashank Murali, Lalka Rieger, Ruoyu Li, Stefan Hommes, Radu State, Gaston Ormazabal, and Henning Schulzrinne. Hanzo: Collaborative network defense for connected things. In *2018 Principles, Systems and Applications of IP Telecommunications (IPTComm)*, pages 1–8. IEEE, 2018.

[91] Soteris Demetriou, Nan Zhang, Yeonjoon Lee, XiaoFeng Wang, Carl A Gunter, Xiaoyong Zhou, and Michael Grace. Hanguard: Sdn-driven protection of smart home wifi devices from malicious mobile apps. In *Proceedings of the 10th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, pages 122–133, 2017.

[92] Vincentius Martin, Qiang Cao, and Theophilus Benson. Fending off iot-hunting attacks at home networks. In *Proceedings of the 2nd Workshop on Cloud-Assisted Networking*, pages 67–72, 2017.

[93] Gunes Acar, Danny Yuxing Huang, Frank Li, Arvind Narayanan, and Nick Feamster. Web-based attacks to discover and control local iot devices. In *Proceedings of the 2018 Workshop on IoT Security and Privacy*, pages 29–35, 2018.

[94] Sanket Goutam, William Enck, and Bradley Reaves. Hestia: simple least privilege network policies for smart homes. In *Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks*, pages 215–220, 2019.

[95] Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael K. Reiter. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In *Proc. CCS*, 2016.

[96] Nirupam Roy, Sheng Shen, Haitham Hassanieh, and Romit Roy Choudhury. Inaudible voice commands: The long-range attack and defense. In *Proc. NSDI*, pages 547–560, 2018.

[97] Takeshi Sugawara, Benjamin Cyr, Sara Rampazzi, Daniel Genkin, and Kevin Fu. Light commands: Laser-based audio injection attacks on voice-controllable systems. In *Proc. USENIX Security*, pages 2631–2648, 2020.

[98] Apple Inc. Play audio on homepod using an ios or ipad os device. `https://support.apple.com/guide/homepod/play-audio-using-your-ios-or-ipados-device-apdfb81a72e4/1.0/homepod/1.0`.

[99] Dan Simmons. BBC fools HSBC voice recognition security system. BBC, May 2017. `https://www.bbc.com/news/technology-39965545`.

[100] Linghan Zhang, Sheng Tan, and Jie Yang. Hearing your voice is not enough: An articulatory gesture based liveness detection for voice authentication. In *Proc. CCS*, 2017.

[101] Blase Ur, Elyse McManus, Melwyn Pak Yong Ho, and Michael L. Littman. Practical trigger-action programming in the smart home. In *Proc. CHI*, 2014.

[102] Qi Wang, Pubali Datta, Wei Yang, Si Liu, Adam Bates, and Carl A. Gunter. Charting the atack surface of trigger-action IoT platforms. In *Proc. CCS*, 2019.

[103] Lefan Zhang, Weijia He, Jesse Martinez, Noah Brackenbury, Shan Lu, and Blase Ur. AutoTap: Synthesizing and repairing trigger-action programs using LTL properties. In *Proc. ICSE*, 2019.

[104] Samsung. SmartThings: Add a Little Smartness to Your Things, August 2014. `https://www.smartthings.com`, as of June 27, 2022.

[105] Amazon. Echo, November 2014. `https://www.amazon.com/echo`, as of June 27, 2022.

[106] Rayoung Yang and Mark W. Newman. Learning from a Learning Thermostat: Lessons for Intelligent Systems for the Home. In *Proc. UbiComp*, 2013.

[107] Belkin. WeMo Home Automation, January 2012. `https://www.belkin.com/wemo`, as of June 27, 2022.

[108] Philips. Hue, October 2012. `https://www.meethue.com`, as of June 27, 2022.

[109] Earlence Fernandes, Amir Rahmati, Kevin Eykholt, and Atul Prakash. Internet of Things Security Research: A Rehash of Old Ideas or New Intellectual Challenges? *IEEE Security & Privacy*, 15(4):79–84, 2017.

[110] Joseph Bonneau, Cormac Herley, Paul C. Van Oorschot, and Frank Stajano. The Quest to Replace Passwords: A Framework for Comparative Evaluation of Web Authentication Schemes. In *Proc. IEEE SP*, 2012.

[111] Tara Matthews, Kathleen O'Leary, Anna Turner, Manya Sleeper, Jill Palzkill Woelfer, Martin Shelton, Cori Manthorne, Elizabeth F. Churchill, and Sunny Consolvo. Stories from Survivors: Privacy & Security Practices when Coping with Intimate Partner Abuse. In *Proc. CHI*, 2017.

[112] Samsung. SmartThings: Capabilities Reference, January 2018. `https://smartthings.developer.samsung.com/develop/api-ref/capabilities.html`, as of June 27, 2022.

[113] Google. Set up Voice Match on Google Home, October 2017. `https://support.google.com/googlehome/answer/7323910`, as of June 27, 2022.

[114] Mike Prospero. Best Smart Home Gadgets of 2018, January 2018. `https://www.tomsguide.com/us/best-smart-home-gadgets,review-2008.html`, as of June 27, 2022.

[115] Google. Jacquard Powered Smart Jackets, September 2017. `https://atap.google.com/jacquard/`, as of June 27, 2022.

[116] Suman Jana, Arvind Narayanan, and Vitaly Shmatikov. A Scanner Darkly: Protecting User Privacy from Perceptual Applications. In *Proc. IEEE SP*, 2013.

[117] Shrirang Mare, Andrés Molina-Markham, Cory Cornelius, Ronald Peterson, and David Kotz. ZEBRA: Zero-Effort Bilateral Recurring Authentication. In *Proc. IEEE SP*, 2014.

[118] Otto Huhta, Swapnil Udar, Mika Juuti, Prakash Shrestha, Nitesh Saxena, and N. Asokan. Pitfalls in Designing Zero-Effort Deauthentication: Opportunistic Human Observation Attacks. In *Proc. NDSS*, 2016.

[119] Jason I. Hong and James A. Landay. An architecture for privacy-sensitive ubiquitous computing. In *Proc. MobiSys*, 2004.

[120] Yuanying Chen, Wei Dong, Yi Gao, Xue Liu, and Tao Gu. Rapid: A multimodal and device-free approach using noise estimation for robust person identification. *IMWUT*, 1(3), 2017.

[121] Shaunak Mishra, Yasser Shoukry, Nikhil Karamchandani, Suhas N. Diggavi, and Paulo Tabuada. Secure state estimation against sensor attacks in the presence of noise. *IEEE TCNS*, 4(1):49–59, 2017.

[122] Mingmin Zhao, Tianhong Li, Mohammad Abu Alsheikh, Yonglong Tian, Hang Zhao, Antonio Torralba, and Dina Katabi. Through-wall human pose estimation using radio signals. In *Proc. CVPR*, 2018.

[123] Chen-Yu Hsu, Rumen Hristov, Guang-He Lee, Mingmin Zhao, and Dina Katabi. Enabling identification and behavioral sensing in homes using radio reflections. In *Proc. CHI*, 2019.

[124] Ju Wang, Hongbo Jiang, Jie Xiong, Kyle Jamieson, Xiaojiang Chen, Dingyi Fang, and Binbin Xie. Lifs: Low human-effort, device-free localization with fine-grained subcarrier information. In *Proc. MobiCom*, 2016.

[125] Tsutomu Matsumoto, Hiroyuki Matsumoto, Koji Yamada, and Satoshi Hoshino. Impact of artificial "gummy" fingers on fingerprint systems. *OSCDT*, 2002.

[126] A.J. Bernheim Brush, Bongshin Lee, Ratul Mahajan, Sharad Agarwal, Stefan Saroiu, and Colin Dixon. Home automation in the wild: Challenges and opportunities. In *Proc. CHI*, 2011.

[127] Diana Freed, Jackeline Palmer, Diana Minchala, Karen Levy, Thomas Ristenpart, and Nicola Dell. "A stalker's paradise": How intimate partner abusers exploit technology. In *Proc. CHI*, 2018.

[128] Edoardo Maggio. The facial recognition on Samsung's Galaxy Note 8 can be fooled with a photo. Business Insider, 2017. `https://www.businessinsider.com/samsung-galaxy-note-8-facial-recognition-tricked-with-a-photo-2017-9`.

[129] Huan Feng, Kassem Fawaz, and Kang G. Shin. Continuous authentication for voice assistants. In *Proc. MobiCom*, 2017.

[130] Dibya Mukhopadhyay, Maliheh Shirvanian, and Nitesh Saxena. All your voices are belong to us: Stealing voices to fool humans and machines. In *Proc. ESORICS*, 2015.

[131] Hadi Abdullah, Washington Garcia, Christian Peeters, Patrick Traynor, Kevin R. B. Butler, and Joseph Wilson. Practical hidden voice attacks against speech and speaker recognition systems. In *Proc. NDSS*, 2019.

[132] Lea Schönherr, Katharina Kohls, Steffen Zeiler, Thorsten Holz, and Dorothea Kolossa. Adversarial attacks against automatic speech recognition systems via psychoacoustic hiding. In *Proc. NDSS*, 2019.

[133] Denis Foo Kune, John D. Backes, Shane S. Clark, Daniel B. Kramer, Matthew R. Reynolds, Kevin Fu, Yongdae Kim, and Wenyuan Xu. Ghost talk: Mitigating EMI signal injection attacks against analog sensors. In *Proc. IEEE S&P*, 2013.

[134] Zhijing Li, Zhujun Xiao, Yanzi Zhu, Irene Pattarachanyakul, Ben Y. Zhao, and Haitao Zheng. Adversarial localization against wireless cameras. In *Proc. HotMobile*, 2018.

[135] Timothy Trippel, Ofir Weisse, Wenyuan Xu, Peter Honeyman, and Kevin Fu. WALNUT: Waging doubt on the integrity of MEMS accelerometers with acoustic injection attacks. In *Proc. EuroS&P*, 2017.

[136] Sarah Underwood. Distinguishing identical twins. CACM, April 2018.

[137] Ry Crist. Amazon and Google are listening to your voice recordings. Here's what we know about that. CNET, July 2019. `https://www.cnet.com/how-to/amazon-and-google-are-listening-to-your-voice-recordings-heres-what-we-know/`.

[138] Alex Hern. Uber employees 'spied on ex-partners, politicians and Beyoncé', December 2016. `https://www.theguardian.com/technology/2016/dec/13/uber-employees-spying-ex-partners-politicians-beyonce`.

[139] G. Qian, J. Zhang, and A. Kidané. People identification using floor pressure sensing and analysis. *IEEE Sensors Journal*, 10(9):1447–1460, 2010.

[140] NETATMO. Smart indoor camera. `https://www.netatmo.com/en-us/security/cam-indoor/specifications`.

[141] Honeywell. 5853 Wireless Glassbreak Detector . `https://www.security.honeywell.com/product-repository/5853`.

[142] Reham Mohamed and Moustafa Youssef. Heartsense: Ubiquitous accurate multi-modal fusion-based heart rate estimation using smartphones. *IMWUT*, 1(3), September 2017.

[143] Yang Zhang, Chouchang Yang, Scott E. Hudson, Chris Harrison, and Alanson P. Sample. Wall++: Room-scale interactive and context-aware sensing. In *Proc. CHI*, 2018.

[144] Gierad Laput, Karan Ahuja, Mayank Goel, and Chris Harrison. Ubicoustics: Plug-and-play acoustic activity recognition. In *Proc. UIST*, 2018.

[145] Avinash Kalyanaraman, Dezhi Hong, Elahe Soltanaghaei, and Kamin Whitehouse. FormaTrack: Tracking people based on body shape. *IMWUT*, 1(3), 2017.

[146] Chi-Jui Wu, Steven Houben, and Nicolai Marquardt. Eaglesense: Tracking people and devices in interactive spaces using real-time top-view depth-sensing. In *Proc. CHI*, 2017.

[147] Robert J. Orr and Gregory D. Abowd. The smart floor: A mechanism for natural user identification and tracking. In *Proc. CHI EA*, 2000.

[148] Kun Qian, Chenshu Wu, Yi Zhang, Guidong Zhang, Zheng Yang, and Yunhao Liu. Widar2.0: Passive human tracking with a single Wi-Fi link. In *Proc. MobiSys*, 2018.

[149] Tianxing Li, Qiang Liu, and Xia Zhou. Practical human sensing in the light. In *Proc. MobiSys*, 2016.

[150] Souvik Sen, Dongho Kim, Stephane Laroche, Kyu-Han Kim, and Jeongkeun Lee. Bringing CUPID indoor positioning system to practice. In *Proc. WWW*, 2015.

[151] Apple. About Touch ID advanced security technology, 2017. `https://support.apple.com/en-us/HT204587`.

[152] Nicholas D. Lane, Petko Georgiev, Cecilia Mascolo, and Ying Gao. Zoe: A cloud-less dialog-enabled continuous sensing wearable exploiting heterogeneous computation. In *Proc. MobiSys*, 2015.

[153] Jagmohan Chauhan, Yining Hu, Suranga Seneviratne, Archan Misra, Aruna Seneviratne, and Youngki Lee. BreathPrint: Breathing acoustics-based user authentication. In *Proc. MobiSys*, 2017.

[154] Apple. Use Face ID on your iPhone or iPad Pro. `https://support.apple.com/en-us/HT208109`.

[155] Microsoft. Windows Hello: Discover facial recognition on Windows 10. `https://www.microsoft.com/en-us/windows/windows-hello`.

[156] Shaxun Chen, Amit Pande, and Prasant Mohapatra. Sensor-assisted facial recognition: an enhanced biometric authentication system for smartphones. In *Proc. MobiSys*, 2014.

[157] Samsung. Galaxy S8 — S8+ - Security. `https://www.samsung.com/global/galaxy/galaxy-s8/security/`.

[158] Aditya Singh Rathore, Weijin Zhu, Afee Daiyan, Chenhan Xu, Kun Wang, Feng Lin, Kui Ren, and Wenyao Xu. Sonicprint: A generally adoptable and secure fingerprint biometrics in smart devices. In *Proc. MobiSys*, 2020.

[159] Cory Cornelius, Ronald Peterson, Joseph Skinner, Ryan Halter, and David Kotz. A wearable system that knows who wears it. In *Proc. MobiSys*, 2014.

[160] Munehiko Sato, Rohan S. Puri, Alex Olwal, Yosuke Ushigome, Lukas Franciszkiewicz, Deepak Chandra, Ivan Poupyrev, and Ramesh Raskar. Zensei: Embedded, multi-electrode bioimpedance sensing for implicit, ubiquitous user recognition. In *Proc. CHI*, 2017.

[161] Feng Lin, Chen Song, Yan Zhuang, Wenyao Xu, Changzhi Li, and Kui Ren. Cardiac scan: A non-contact and continuous heart-based user authentication system. In *Proc. MobiCom*, 2017.

[162] Jian Liu, Cong Shi, Yingying Chen, Hongbo Liu, and Marco Gruteser. Cardiocam: Leveraging camera on mobile devices to verify users while their heart is pumping. In *Proc. MobiSys*, 2019.

[163] Juhi Ranjan and Kamin Whitehouse. Object hallmarks: Identifying object users using wearable wrist sensors. In *Proc. UbiComp*, 2015.

[164] Shijia Pan, Tong Yu, Mostafa Mirshekari, Jonathon Fagert, Amelie Bonde, Ole J. Mengshoel, Hae Young Noh, and Pei Zhang. FootprintID: Indoor pedestrian identification through ambient structural vibration sensing. *IMWUT*, 1(3), 2017.

[165] Liang Wang, Tieniu Tan, Huazhong Ning, and Weiming Hu. Silhouette analysis-based gait recognition for human identification. *IEEE TPAMI*, 25(12), 2003.

[166] Jaeseok Yun. User identification using gait patterns on UbiFloorII. *Sensors*, 11(3), 2011.

[167] Mohammad Sedaaghi. A comparative study of gender and age classification in speech signals. *IJEEE*, 5, 03 2009.

[168] Dat Tien Nguyen, So Ra Cho, Tuyen Danh Pham, and Kang Ryoung Park. Human age estimation method robust to camera sensor and/or face movement. *Sensors*, 15(9), 2015.

[169] Yu Zhang and Dit-Yan Yeung. Multi-task warped Gaussian process for personalized age estimation. In *Proc. CVPR*, 2010.

[170] Hongyu Pan, Hu Han, Shiguang Shan, and Xilin Chen. Mean-variance loss for deep age estimation from a face. In *Proc. CVPR*, 2018.

[171] Hamdi Dibeklioglu, Fares Alnajar, Albert Ali Salah, and Theo Gevers. Combining facial dynamics with appearance for age estimation. *IEEE TIP*, 24(6), 2015.

[172] Google Nest. Split-spectrum white paper. Technical report, June 2015.

[173] First Alert. Battery powered photo & ion smoke alarm. `https://images-na.ssl-images-amazon.com/images/I/A1+UJjI+uPL.pdf`.

[174] B. Ugur Töreyin, R. Gokberk Cinbis, Yigithan Dedeoglu, and A. Enis Cetin. Fire detection in infrared video using wavelet analysis. *Opt. Eng.*, 46(6), 2007.

[175] Sierra Monitor Corporation. `https://www.sierramonitor.com/flame-detector-3600-lb`.

[176] Techamor. `https://www.amazon.com/dp/B07BM1XWB8`.

[177] Google. Google Nest Protect. `https://store.google.com/us/product/nest_protect_2nd_gen`.

[178] Apple. iBeacon. `https://developer.apple.com/ibeacon/`.

[179] Nissanka B. Priyantha, Anit Chakraborty, and Hari Balakrishnan. The Cricket location-support system. In *Proc. MobiCom*, 2000.

[180] James Scott, A.J. Bernheim Brush, John Krumm, Brian Meyers, Michael Hazas, Stephen Hodges, and Nicolas Villar. Preheat: Controlling home heating using occupancy prediction. In *Proc. UbiComp*, 2011.

[181] Kazuya Ohara, Takuya Maekawa, Yasue Kishino, Yoshinari Shirai, and Futoshi Naya. Transferring positioning model for device-free passive indoor localization. In *Proc. UbiComp*, 2015.

[182] Yan Wang, Jian Liu, Yingying Chen, Marco Gruteser, Jie Yang, and Hongbo Liu. E-eyes: Device-free location-oriented activity identification using fine-grained WiFi signatures. In *Proc. MobiCom*, 2014.

[183] Fan Li, Chunshui Zhao, Guanzhong Ding, Jian Gong, Chenxing Liu, and Feng Zhao. A reliable and accurate indoor localization method using phone inertial sensors. In *Proc. Ubicomp*, 2012.

[184] Gabriele Civitarese, Stefano Belfiore, and Claudio Bettini. Let the objects tell what you are doing. In *Proc. UbiComp*, 2016.

[185] Yang Zhang, Yasha Iravantchi, Haojian Jin, Swarun Kumar, and Chris Harrison. Sozu: Self-powered radio tags for building-scale activity sensing. In *Proc. UIST*, 2019.

[186] Roy Want, Andy Hopper, Veronica Falcão, and Jonathan Gibbons. The active badge location system. *ACM TIS*, 10(1):91–102, 1992.

[187] Patrick Lazik and Anthony Rowe. Indoor pseudo-ranging of mobile devices using ultrasonic chirps. In *Proc. SenSys*, 2012.

[188] Mike Addlesee, Rupert Curwen, Steve Hodges, Joe Newman, Pete Steggles, Andy Ward, and Andy Hopper. Implementing a sentient computing system. *Computer*, 34(8):50–56, 2001.

[189] Tobias Grosse-Puppendahl, Sebastian Herber, Raphael Wimmer, Frank Englert, Sebastian Beck, Julian von Wilmsdorff, Reiner Wichert, and Arjan Kuijper. Capacitive near-field communication for ubiquitous interaction and perception. In *Proc. UbiComp*, 2014.

[190] Weizhi Zhang, MI Sakib Chowdhury, and Mohsen Kavehrad. Asynchronous indoor positioning system based on visible light communications. *Opt. Eng.*, 53(4), 2014.

[191] NAPCO. Napco adaptive dual microwave/PIR detector, 30x35 ft. (c-100ste). `https://www.amazon.com/Napco-Adaptive-Microwave-Detector-C-100STE/dp/B0041X47EW`.

[192] Sidhant Gupta, Ke-Yu Chen, Matthew S. Reynolds, and Shwetak N. Patel. Lightwave: Using compact fluorescent lights as sensors. In *Proc. UbiComp*, 2011.

[193] Edward J. Wang, Tien-Jui Lee, Alex Mariakakis, Mayank Goel, Sidhant Gupta, and Shwetak N. Patel. Magnifisense: Inferring device interaction using wrist-worn passive magneto-inductive sensors. In *Proc. UbiComp*, 2015.

[194] Tobias Grosse-Puppendahl, Xavier Dellangnol, Christian Hatzfeld, Biying Fu, Mario Kupnik, Arjan Kuijper, Matthias R. Hastall, James Scott, and Marco Gruteser. Platypus - Indoor localization and identification through sensing electric potential changes in human bodies. In *Proc. MobiSys*, 2016.

[195] Edison Thomaz, Vinay Bettadapura, Gabriel Reyes, Megha Sandesh, Grant Schindler, Thomas Plötz, Gregory D. Abowd, and Irfan Essa. Recognizing water-based activities in the home through infrastructure-mediated sensing. In *Proc. UbiComp*, 2012.

[196] National Coordination Office for Space-Based Positioning, Navigation, and Timing. GPS: The Global Positioning System. `https://www.gps.gov/`.

[197] Bharathan Balaji, Jian Xu, Anthony Nwokafor, Rajesh Gupta, and Yuvraj Agarwal. Sentinel: Occupancy based HVAC actuation using existing WiFi infrastructure within commercial buildings. In *Proc. SenSys*, 2013.

[198] Google Nest Cam. Indoor - tech specs. `https://store.google.com/us/product/nest_cam_specs`.

[199] Alan Bränzel, Christian Holz, Daniel Hoffmann, Dominik Schmidt, Marius Knaust, Patrick Lühne, René Meusel, Stephan Richter, and Patrick Baudisch. GravitySpace: Tracking users and their poses in a smart room using a pressure-sensing floor. In *Proc. CHI EA*, 2013.

[200] Timothy W. Hnat, Erin Griffiths, Raymond Dawson, and Kamin Whitehouse. Doorjamb: Unobtrusive room-level tracking of people in homes using doorway sensors. In *Proc. SenSys*, 2012.

[201] Samsung. Motion sensor. `https://www.lowes.com/pd/Samsung-Motion-Sensor/1000555661`.

[202] NAPCO. Napco's adaptive dual microwave/PIR detectors automatically adjust to their environment, minute by minute, for the ultimate false alarm immunity & reliability. `https://napcosecurity.com/products/napco-detectors/`.

[203] Honeywell. DT906 / DT907. `https://www.security.honeywell.com/product-repository/dt906-dt907`.

[204] CO2Meter. Tim10 desktop co2, temp. & humidity monitor. `https://www.co2meter.com/products/tim10-desktop-co2-temp-humidity-monitor`.

[205] GridEye. Infrared Array Sensor Grid-EYE: High Precision Infrared Array Sensor based on Advanced MEMS Technology. `https://www.mouser.com/datasheet/2/315/ADI8000C65-1267019.pdf`.

[206] Apple. Healthkit. `https://developer.apple.com/healthkit/`.

[207] Fitbit Inc. How do I track my activity with my Fitbit device? `https://help.fitbit.com/articles/en_US/Help_article/1785`.

[208] Tauhidur Rahman, Alexander T. Adams, Ruth Vinisha Ravichandran, Mi Zhang, Shwetak N. Patel, Julie A. Kientz, and Tanzeem Choudhury. Dopplesleep: A contactless unobtrusive sleep sensing system using short-range doppler radar. In *Proc. UbiComp*, 2015.

[209] Adiyan Mujibiya and Jun Rekimoto. Mirage: Exploring interaction modalities using off-body static electric field sensing. In *Proc. UIST*, 2013.

[210] Sheng Tan, Linghan Zhang, Zi Wang, and Jie Yang. Multitrack: Multi-user tracking and activity recognition using commodity wifi. In *Proc. CHI*, 2019.

[211] Di Tang, Zhe Zhou, Yinqian Zhang, and Kehuan Zhang. Face Flashing: A secure liveness detection protocol based on light reflections. In *Proc. NDSS*, 2018.

[212] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Florian Tramèr, Atul Prakash, Tadayoshi Kohno, and Dawn Song. Physical adversarial examples for object detectors. In *Proc. WOOT*, 2018.

[213] Chun-Chen Tu, Pai-Shun Ting, Pin-Yu Chen, Sijia Liu, Huan Zhang, Jinfeng Yi, Cho-Jui Hsieh, and Shin-Ming Cheng. Autozoom: Autoencoder-based zeroth order optimization method for attacking black-box neural networks. In *Proc. AAAI*, 2019.

[214] Andrew Ilyas, Logan Engstrom, and Aleksander Madry. Prior convictions: Black-box adversarial attacks with bandits and priors. In *Proc. ICLR*, 2019.

[215] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into transferable adversarial examples and black-box attacks. In *Proc. ICLR*, 2017.

[216] Fnu Suya, Jianfeng Chi, David Evans, and Yuan Tian. Hybrid batch attacks: Finding black-box adversarial examples with limited queries. In *Proc. USENIX Security*, 2020.

[217] Jakub Konečnỳ, H. Brendan McMahan, Daniel Ramage, and Peter Richtárik. Federated optimization: Distributed machine learning for on-device intelligence. arXiv:1610.02527, 2016.

[218] Aditya Kusupati, Manish Singh, Kush Bhatia, Ashish Kumar, Prateek Jain, and Manik Varma. FastGRNN: A fast, accurate, stable and tiny kilobyte sized gated recurrent neural network. In *Proc. NeurIPS*, 2018.

[219] Sridhar Gopinath, Nikhil Ghanathe, Vivek Seshadri, and Rahul Sharma. Compiling KB-sized machine learning models to tiny IoT devices. In *Proc. PLDI*, 2019.

[220] Nisarg Raval, Ashwin Machanavajjhala, and Jerry Pan. Olympus: Sensor privacy through utility aware obfuscation. *PoPETS*, 2019(1):5–25, 2019.

[221] Pardis Emami Naeini, Yuvraj Agarwal, Lorrie Faith Cranor, and Hanan Hibshi. Ask the experts: What should be on an IoT privacy and security label? In *Proc. IEEE S&P*, 2020.

[222] Serge Egelman, Raghudeep Kannavara, and Richard Chow. Is this thing on?: Crowdsourcing privacy indicators for ubiquitous sensing platforms. In *Proc. CHI*, 2015.

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

[224] 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 *Proc. USENIX Security*, pages 1093–1110, 2017.

[225] 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.

[226] Anna Kornfeld Simpson, Franziska Roesner, and Tadayoshi Kohno. Securing vulnerable home iot devices with an in-hub security manager. In *2017 IEEE International Conference on Pervasive Computing and Communications Workshops*, pages 551–556. IEEE, 2017.

[227] Naman Gupta, Vinayak Naik, and Srishti Sengupta. A firewall for internet of things. In *9th International Conference on Communication Systems and Networks*, pages 411–412. IEEE, 2017.

[228] Anonymous. Code and artifacts, 2022.

[229] Synology Inc. What network ports are used by synology services? `https://kb.synology.com/en-global/DSM/tutorial/What_network_ports_are_used_by_Synology_services`, 2022.

[230] Wikipedia. Cingular wireless. `https://simple.wikipedia.org/wiki/Cingular_Wireless`, 2022.

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

[232] PubNub Inc. Logitech powers smart home hub, remote control, and app using pubnub. `https://www.pubnub.com/customers/logitech/`, 2022.

[233] Wikipedia. Ring (company) - wikipedia, 2022. `https://en.wikipedia.org/wiki/Ring_(company)`.

[234] 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`.

CHAPTER 7

APPENDIX

7.1 Home IoT Devices Considered in Chapter 4

Cooking Devices	Anova Culinary Precision Cooker Char-Broil Digital Electric Smoker June Intelligent Oven Perfect Bake Pro Samsung Family Refrigerator
Hubs	Samsung SmartThings Wink Hub 2
Lights/Power Plugs	Belkin Wemo Insight Switch BeOn iHome Smartplug LIFX Color 1000 Lutron Caseta In-Wall Wireless Lighting Philips Hue Starter Set
Locks	August SmartLock Kwikset Smartcode Touchscreen
Outdoor Devices	Rachio Smart Sprinkler Robomow
Security Cameras	Kuna Toucan LG Smart Security Wireless Camera Nest Cam NetGear ArloPro Skybell Video Doorbell Tend Secure Lynx Indoor
Thermostats	EcoBee 4 Hisense Portable AC Nest Learning Thermostat
Voice Assistants	Amazon Echo Echo Dot Google Home

Table 7.1: Home IoT devices evaluated for Chapter 4.

7.2 Full Descriptions of Capabilities for the Survey in Chapter 4

- **Software Update**: Install a software update to get the latest features, improvements, and security updates.
- **Play Music**: Play music (e. g., from Spotify) in the house.
- **Order Online**: Make online purchases (e. g., on Amazon) on a shared household account.
- **Temperature Log**: View the last 10 temperature adjustments and who made them.
- **Mower On/Off**: Turn the lawn mower on or off remotely (i. e., on a smartphone, from anywhere).
- **Mower Rule**: Create rules that specify what the lawn mower should do, connecting its actions to other devices, sensors, and services. For example, one could create a rule specifying that the mower should not mow if it is raining.
- **Lock Log**: View an activity log for the past week that shows who entered the home at what times. People will be identified based on whose PIN code or smartphone was used to unlock the door.
- **Lock State**: See whether the front door is currently locked or unlocked.
- **Lock Rule**: Create rules that specify when the lock should be locked or unlocked, connecting it to other devices, sensors, and services. For example, one could create a rule specifying that the lock should always be locked when no one is home.
- **Answer Door**: Answer the doorbell by seeing a live video of who is at the front door and having the opportunity to unlock the door remotely (e. g., on a smartphone, from anywhere).
- **Delete Lock Log**: Delete the activity log that records who has tried to open or close the door.
- **Lights State**: See which lights in the home are on or off.
- **Lights On/Off**: Remotely control whether a light is currently on, as well as how

bright it is (e. g., on a smartphone, from anywhere).

- **Lights Rule**: Create rules that specify when the lights should turn on/off or change color based on other sensors, devices, and services. For example, one could create rules specifying how the lights automatically change brightness or color based on the current weather or the movie played on the TV.
- **Light Scheme**: Allow a streaming video provider to change the lighting according to the theme of the movie that is currently being watched.
- **New Device**: Connect a new device to the hub, enabling the hub to control that device.
- **New User**: Add new users (people) to the smart-home management system, as well as remove users from the smart-home management system.
- **Live Video**: See live video from each camera in or around the house.
- **Facial Recognition**: Enable or disable facial recognition technology for a person. This technology is used to identify them automatically in video recordings.
- **Delete Video**: Delete one or more previously recorded videos.
- **Camera On/Off**: Turn the camera on/off remotely (e. g., on a smartphone, from anywhere).
- **Camera Angle**: Change camera's view remotely (including turning its lens to view a different angle, zooming in/out, etc.).

7.3 Full Descriptions of Relationships for the Survey in Chapter 4

- **Your spouse**: Imagine you have a spouse. You live with them everyday and share all smart appliances in your home. You make decisions together in most cases, especially important ones.
- **Your teenage child**: Imagine you have a 16-year-old child. They live with you, go

to school in the morning, and come back in the afternoon (on the weekdays). They are familiar with all of these Smart devices in your home, and enjoy using them. They know how to use these devices as well as you do, if not better. They spend a lot of time on their smartphone. They usually are well-behaved, but they are still a teenager.

- **Your child in elementary school**: Imagine you have an 8-year-old child who is still in elementary school. They live with you and go to school daily, unless it's the weekend or a holiday. They have a basic idea of how to use smart devices. However, they don't know how to use some more complex features properly, like changing the settings, but it doesn't discourage them from trying. They do not have their own smartphone, but they keep asking you for one.
- **A visiting family member**: Imagine you have a visiting family member. They are about the same age as you, if not much older. You grew up together, but now you meet each other once or twice a year, because you live far away from each other. They visit you on holidays or other big events. They usually stay with you for several days, maybe even a little bit past the holiday, and they remain at home alone while you are away for work.
- **The babysitter**: Imagine you have a babysitter in your home for taking care of your child. They will be at your place while you are at work. They work 4 hours after school, 3 days per week. You have known them over 6 months and you are satisfied with their work so far, and have no intention of letting them go anytime soon.
- **Your neighbor**: Imagine you have a neighbor living next to you. You don't know them very well, but they seem to be good people. If you meet them on the street, you greet them and make some friendly small talk. Occasionally you invite them over for dinner, but they are never in your house when you are away.

7.4 Survey Instrument in Chapter 4

Introduction

Computing is transitioning from single-user devices, such as laptops and phones, to the Internet of Things, in which many users will interact with a particular device, such as an Amazon Echo or Internet-connected door lock. Current measures fail to provide usable authentication, access control, or privacy when multiple users share a device. Even more so, the users of a given device often have complex social relationships to each other. Our goal is to develop techniques and interfaces that enable accurate access control and authentication in multi-user IoT environments, based on user preferences.

Participation should take about 20 minutes.

In recent years, many internet-connected (”smart”) home devices and appliances have entered the market. Imagine that you own many such smart devices that are connected both to the Internet and to each other.

This includes a smart hub that can control other devices in your home, particularly with the help of the smart voice assistant. You also have a smart door lock and smart camera for home security, as well as smart lighting and a smart thermostat to control your environment. There is also a smart lawn mower maintaining your lawn. All of these devices can be remotely controlled using a smartphone app by anyone to whom you have given permission. You, or anyone else you have permitted, can also write rules specifying in what situations devices should activate automatically.

In this survey, we will ask you questions about who in your household should be allowed to access one particular feature of a smart device. If you live in multiple places, think of the home in which you live the majority of the time. For all questions, assume that the system has correctly identified the user involved (i. e., there are no cases of mistaken identity).

Because the situations may involve either positive or negative consequences, you should take some time to think about your response. The next button will not appear until you have spent at least 30 seconds on each page.

In this survey, we will ask whether you will allow people of the following relationships to control a particular feature of a smart device: your spouse; your teenage child; your child in elementary school; a visiting family member; a babysitter; your neighbor. Please imagine you have these relationships in your life even if you don't. All of these relationships are separate people.

If you grant access to any of these people, they will be able to access your devices whether or not they are in your home, unless you specify otherwise in your responses in the survey. All questions in this survey will focus on one particular feature, but we will ask about your opinion on how different people should be able to use it.

The following use the example "Your Spouse", a "Smart Hub", and a hub-related capability.

The questions on this page only focus on the following person: **Your spouse**: Imagine you have a spouse. You live with them everyday and share all smart appliances in your home. You make decisions together in most cases, especially important ones.

Imagine you are the owner of a **Smart Hub**.

Should **your spouse** be able to use the following feature? **[capability]** ◯ Always (24/7/365) ◯ Never ◯ Sometimes, depending on specific factors

---

Show questions if "Always" chosen

Why?

Imagine that the device incorrectly denies your spouse the ability to use this feature. How much of an inconvenience, if any, would this be? ◯ Not an inconvenience ◯ Minor inconvenience ◯ Major inconvenience

Why? Please be specific.

Show questions if "Never" chosen

Why?

Imagine that the device incorrectly allows your spouse the ability to use this feature. How much of an inconvenience, if any, would this be?

◯ Not an inconvenience ◯ Minor inconvenience ◯ Major inconvenience

Why? Please be specific.

Show questions if "Sometimes" chosen

When should they be allowed to use this feature? Please be specific.

How important is it that they be allowed to use the feature in the cases you specified above?

◯ Not important ◯ Slightly important ◯ Moderately important ◯ Very important ◯ Extremely important

In contrast, when should they not be allowed to use this feature? Please be specific.

How important is it that they not be allowed to use the feature in the cases you specified above?

◯ Not important ◯ Slightly important ◯ Moderately important ◯ Very important ◯ Extremely important

---

Thanks! We will now be asking you an additional set of questions. Imagine that you have already chosen settings specifying who can and cannot access a certain feature in your home. Think broadly about all types of people you might want to allow to control these devices; do not restrict yourself just to the relationships we have previously asked about.

Scenario: Imagine you are still the owner of a **Smart Hub**. You specify that certain people can access the following feature only sometimes: **[capability]**

Might the **location of the person relative to the device** (e. g., in the same room, not in the house, etc.) affect your decision on whether certain people can or cannot use this particular feature? ◯ Yes ◯ No ◯ Not applicable

Briefly explain your response.

Might the **location of the device in the house** (e. g., which room) affect your decision on whether certain people can or cannot use this particular feature? ◯ Yes ◯ No ◯ Not applicable

Briefly explain your response.

Might the **current state of the device** (e. g., whether it is on or off) affect your decision on whether certain people can or cannot use this particular feature? ◯ Yes ◯ No ◯ Not applicable

Briefly explain your response.

Might the **cost of performing that action** (e. g., cost of electricity or other monetary costs of carrying out that action) affect your decision on whether certain people can or cannot use this particular feature? ◯ Yes ◯ No ◯ Not applicable

Briefly explain your response.

Might the **person's recent usage of the device** affect your decision on whether certain people can or cannot use this particular feature? ◯ Yes ◯ No ◯ Not applicable

Briefly explain your response.

Might the **age of the person** affect your decision on whether certain people can or cannot use this particular feature? ◯ Yes ◯ No ◯ Not applicable

Briefly explain your response.

Might **who else, if anyone, is currently at home** affect your decision on whether certain people can or cannot use this particular feature? ◯ Yes ◯ No ◯ Not applicable

Briefly explain your response.

Might the **time of day** affect your decision on whether certain people can or cannot use this particular feature? ◯ Yes ◯ No ◯ Not applicable

Briefly explain your response.

Please list any other factors that might affect your decision on whether certain people can or cannot use the following feature: **[capability]**

Do you or anyone in your household own the following devices?
Internet-connected lights? ◯ Yes ◯ No
Internet-connected thermostat? ◯ Yes ◯ No
Internet-connected voice assistant? ◯ Yes ◯ No
Internet-connected lawn mower? ◯ Yes ◯ No
Internet-connected security camera? ◯ Yes ◯ No
Internet-connected door lock? ◯ Yes ◯ No

If answered yes to any of the above: Which specific devices (brand, model, etc.) do you own?

Please choose the answer that best applies:

Spouse: ◯ I'm currently living with such a person ◯ I'm not currently living with such a person, but I have previously ◯ I have never lived with such a person ◯ I prefer not to answer

Child in elementary school: ◯ I'm currently living with such a person ◯ I'm not currently living with such a person, but I have previously ◯ I have never lived with such a person ◯ I prefer not to answer

Teenage child: ◯ I'm currently living with such a person ◯ I'm not currently living with such a person, but I have previously ◯ I have never lived with such a person ◯ I prefer not to answer

Which of the following best describes your experience with hiring a babysitter (someone unrelated to you whom you pay to watch your children)? ◯ I have hired a babysitter within the last year ◯ I have hired a babysitter but not within the last year ◯ I have never hired a babysitter ◯ I prefer not to answer

Which of the following best describes your neighbors? ◯ I have neighbors and I know most of them ◯ I have neighbors and I know some of them ◯ I have neighbors and I know few or none of them ◯ I do not have neighbors ◯ I prefer not to answer

In a typical year, how many nights total do relatives (who do not live with you) stay at your home? ◯ 0 ◯ 1-10 ◯ 10-20 ◯ 20-30 ◯ 30+ ◯ I prefer not to answer

Do you live in a: ◯ Single family home ◯ Townhouse ◯ Apartment/condo ◯ Other (please specify) ◯ I prefer not to answer

Do you rent or own the place where you live? ◯ Rent ◯ Own ◯ I prefer not to answer

How many people (including you) are there in your household? ◯ 1 ◯ 2 ◯ 3 ◯ 4 ◯ 5 ◯ More than 5 ◯ I prefer not to answer

What is your age range? ◯ 18-24 ◯ 25-34 ◯ 35-44 ◯ 45-54 ◯ 55-64 ◯ 65-74 ◯ 75+ ◯ Prefer not to say

With what gender do you identify? ◯ Male ◯ Female ◯ Non-binary ◯ Other _____ ◯ Prefer not to say

Are you majoring in, hold a degree in, or have held a job in any of the following fields: computer science; computer engineering; information technology; or a related field? ◯ Yes ◯ No ◯ Prefer not to answer

If you have any further feedback, questions, comments, concerns, or anything else you want to tell us, please leave a comment below!

7.5 Traffic Throughput by Device Type

Although the paper focuses on the endpoints home IoT devices contact, we also examined the variability in the amount of traffic they send and receive. In cases like the Mirai botnet [224], IoT devices were repurposed for DDoS. Thus, characterizing and perhaps throttling the amount of traffic they could send is an additional network-level defense. **However, we notice that vendors may occasionally request a lot of data from devices and vice versa. Setting up throttling may interfere with a device's legitimate functions.**

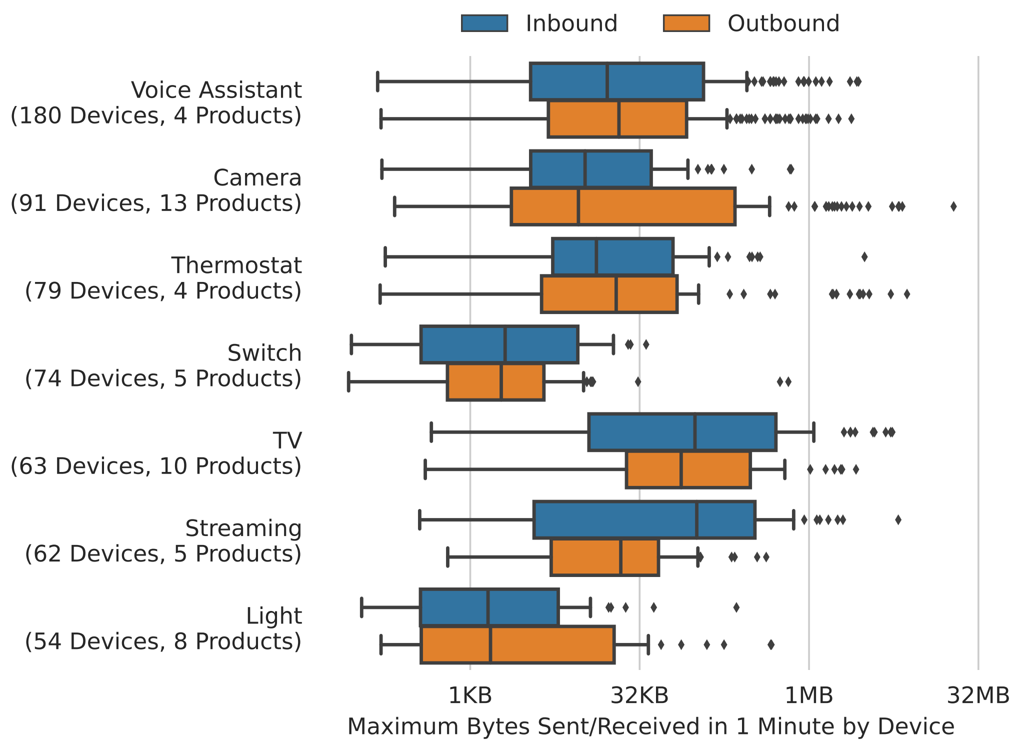


Figure 7.1: Distributions of the maximum bytes that a given device sends (receives) within 1 minute, grouping by type.

Thus, throttling is not included in the body of the paper, but only discussed here for completeness.

We expected that, while products of the same type may not share similarities in the remote hosts they contact, they might require similar throughput, which could imply throttling limits. Figure 7.1 thus shows, for a larger set of products, the maximum amount of traffic any given device in our dataset in any given one-minute period. The figure only includes flows with an MTU ≤ 1500.[1] Figure 7.1 shows that TVs in our dataset received up to 5.5 MB of data per minute, while switches received up to 36.5 KB per minute at most. As expected, switches and lights receive and send much less data than other types. However, we observed some surprising outliers. For example, a TP-Link switch and a Belkin Wemo switch

1. We impose a limit on the MTU as some packet sizes collected by IoT Inspector significantly exceed the MTU, up to millions of bytes per packet. We observe these apparently erroneous MTU values on 2.4% of devices.

respectively sent 671.12 KB and 565.55 KB in one minute to `n-devs.tplinkcloud.com` and `nat.xbcs.net`, respectively. We contemplate that these outliers may be caused by some rare user operations or by the vendors' own data collection. A higher throttling threshold can be placed on the device if one would like to ensure full functionality. On the other hand, TVs and streaming devices generally receive more traffic than others. One interesting observation is that cameras are the only ones that send much more data than receive. The amount of data cameras sent in one minute also varies a lot, from tens of bytes to 19.28 MB, making them harder to throttle.

7.6 Functionalities Tested in the 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.
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.
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.

Table 7.2: Product functionalities tested in our lab evaluation.

Functionality	Description
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 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.
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.

Table 7.3: (Continued) Product functionalities tested in our lab evaluation.

Functionality	Description
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.
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.
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.

Table 7.4: (Continued) Product functionalities tested in our lab evaluation.

7.7 Regionalization Regressions

Factor	β	**SE**	t	p
(Intercept)	0.828	0.004	194.915	$< .001$
Train Region: B	-0.025	0.001	-19.322	$< .001$
Train Region: C	-0.036	0.001	-26.541	$< .001$
Test Region: B	-0.041	0.002	-22.423	$< .001$
Test 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	$< .001$
Train Region: B * Test Region: B	0.060	0.003	23.276	$< .001$
Train Region: B * Test Region: C	0.063	0.005	12.131	$< .001$
Train Region: C * Test Region: B	0.054	0.003	21.339	$< .001$
Train Region: C * Test Region: C	0.128	0.007	19.644	$< .001$

Table 7.5: Linear regressions modeling the impact on **hostname-based allowlists** transferability of the geographic region in which devices were located, whether the train and test regions were the same (* represents an interaction term), the amount of training data, and the product. As the baseline for categorical variables, we use the largest category: *A* for region and *Amazon Echo* for product.

Factor	***β***	**SE**	***t***	***p***
(Intercept)	0.792	0.004	213.467	$< .001$
Train Region: B	-0.031	0.001	-25.069	$< .001$
Train Region: C	-0.032	0.001	-25.500	$< .001$
Test Region: B	-0.036	0.002	-21.423	$< .001$
Test 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	$< .001$
Train Region: B * Test Region: B	0.074	0.002	30.675	$< .001$
Train Region: B * Test Region: C	0.055	0.005	11.369	$< .001$
Train Region: C * Test Region: B	0.064	0.002	27.024	$< .001$
Train Region: C * Test Region: C	0.121	0.006	19.944	$< .001$

Table 7.6: Linear regressions modeling the impact on **hostname-pattern-based allowlists** transferability of the geographic region in which devices were located, whether the train and test regions were the same (* represents an interaction term), the amount of training data, and the product. As the baseline for categorical variables, we use the largest category: *A* for region and *Amazon Echo* for product.

Factor	***β***	**SE**	***t***	***p***
(Intercept)	0.915	0.004	225.965	$< .001$
Train Region: C	-0.034	0.001	-30.047	$< .001$
Train Region: B	-0.013	0.001	-11.484	$< .001$
Test Region: C	-0.053	0.003	-17.047	$< .001$
Test 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	$< .001$
Train Region: C * Test Region: C	0.092	0.006	16.720	$< .001$
Train Region: B * Test Region: C	0.036	0.004	8.231	$< .001$
Train Region: C * Test Region: B	0.044	0.002	20.333	$< .001$
Train Region: B * Test Region: B	0.022	0.002	10.072	$< .001$

Table 7.7: Linear regressions modeling the impact on **domain-based allowlists** transferability of the geographic region in which devices were located, whether the train and test regions were the same (* represents an interaction term), the amount of training data, and the product. As the baseline for categorical variables, we use the largest category: *A* for region and *Amazon Echo* for product.

7.8 Example Allowlist

```
{
  "device_name": "Amazon Echo in Study",
  "device_mac": "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-p53l.ap.spotify.com": "(guc|gae|gew)[0-9]+-
accesspoint-.*\.ap\.spotify\.com",
      "spectrum.s3.amazonaws.com": "spectrum\.s3\.amazonaws\.com",
      ...
    }
  }
}
```

Listing 7.1: Example of our allowlist format.

7.9 Additional Figures

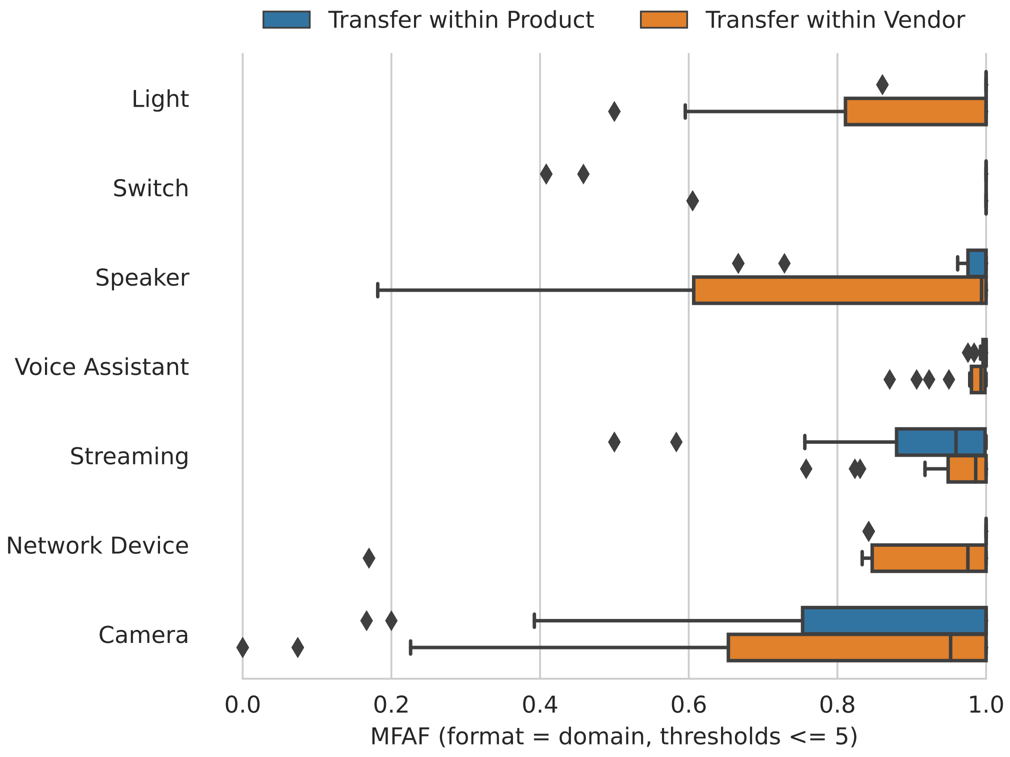


Figure 7.2: A comparison of the transferability of allowlists within a given product and across products from the same vendor, grouped by type. Each point in a boxplot is the vendor MFAF of a product of the specified type. The plot uses domain-based allowlists with threshold ≤ 5.

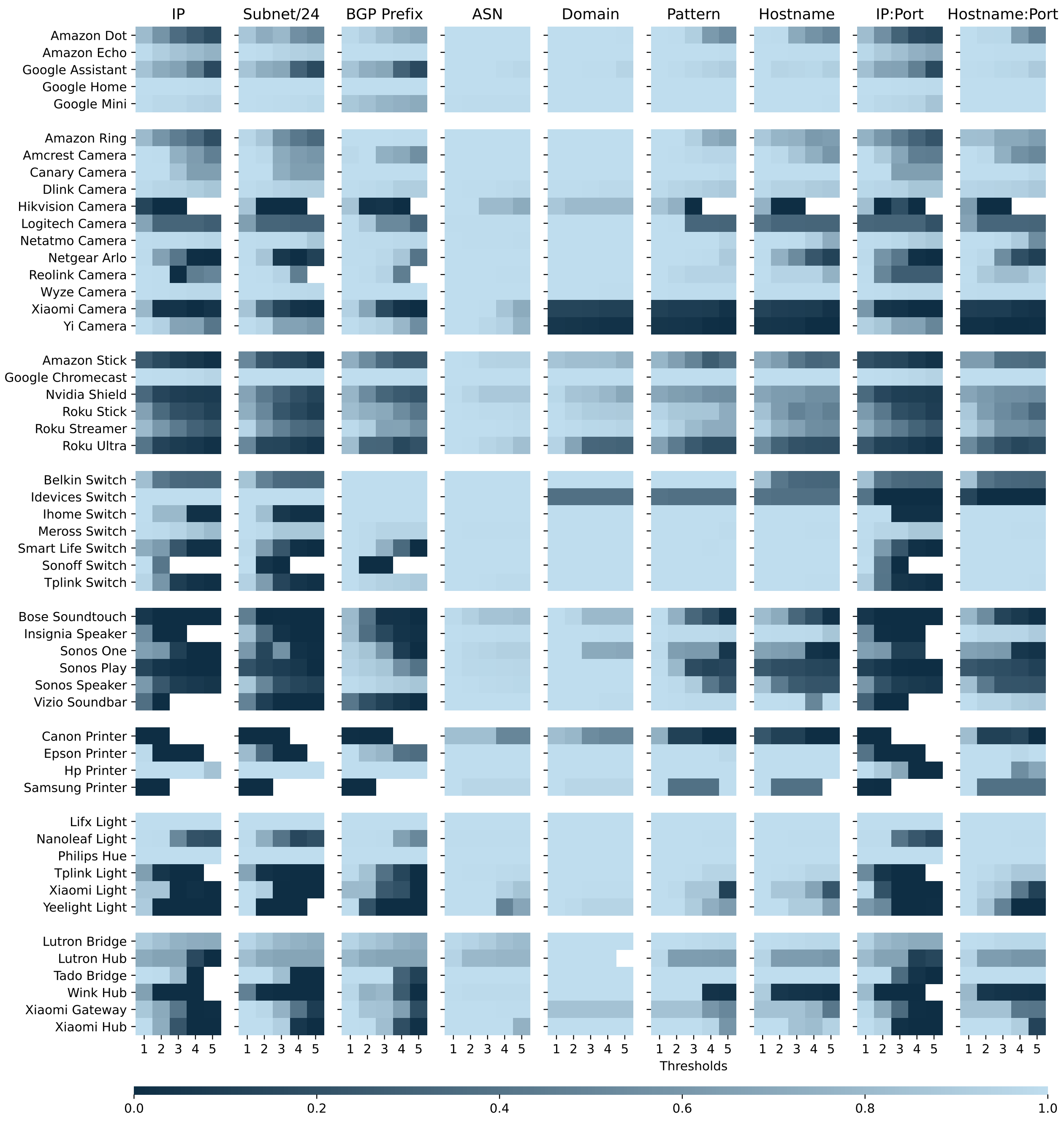


Figure 7.3: MFAF of device network behavior using different allowlist formats (columns). Rows represent the source product for generating the allowlists, and the color indicates the proportion of flows allowed when creating an allowlist using a threshold of between 1 and 5s (sub-columns) and applying it to devices of **the same product**.

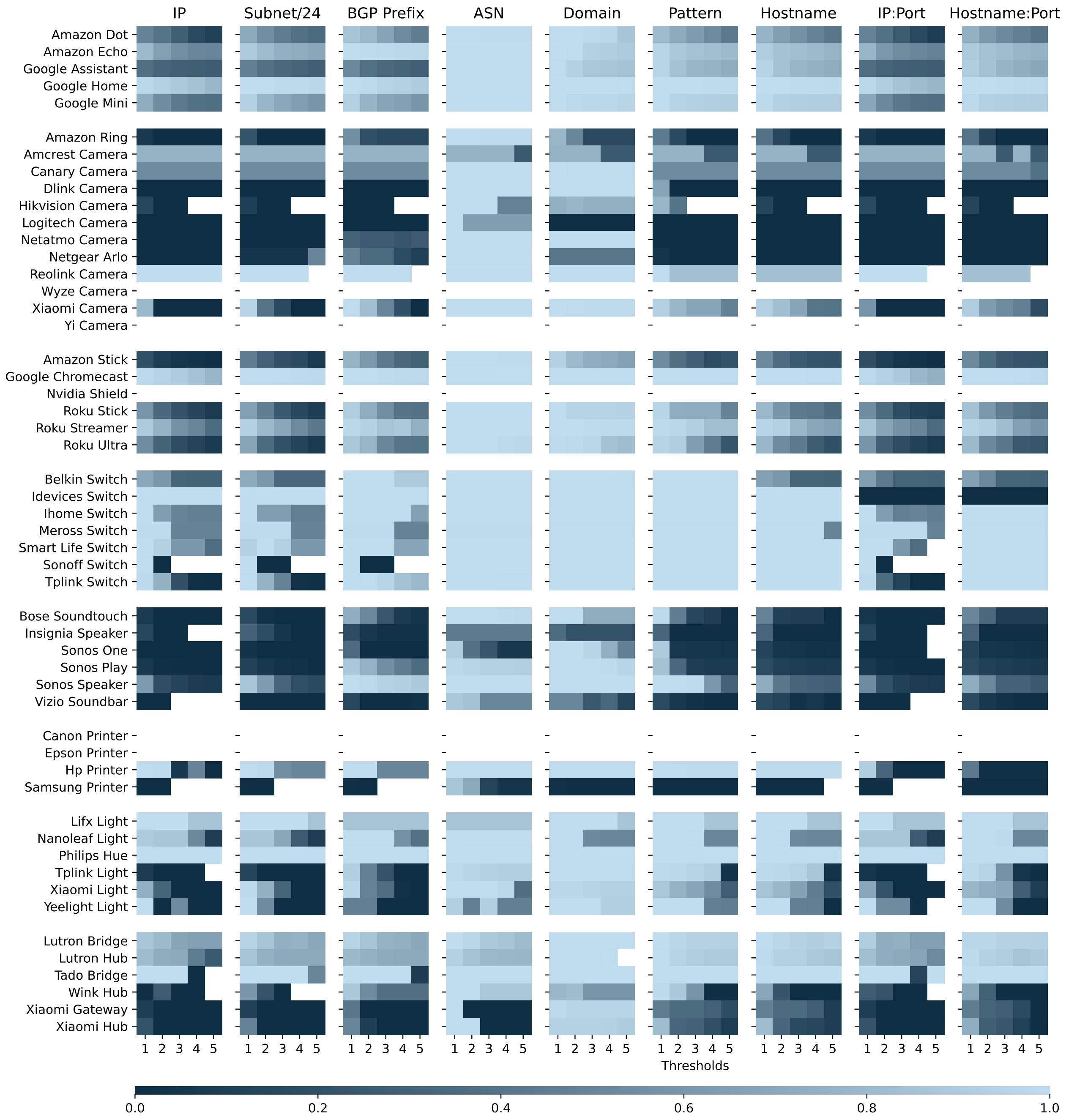


Figure 7.4: MFAF of allowlists using different host representations (columns). Rows represent the source product for generating the allowlists, and the color indicates the proportion of flows allowed when creating an allowlist using a threshold of between 1 and 5s (sub-columns) and applying it to devices of **the same vendor**.

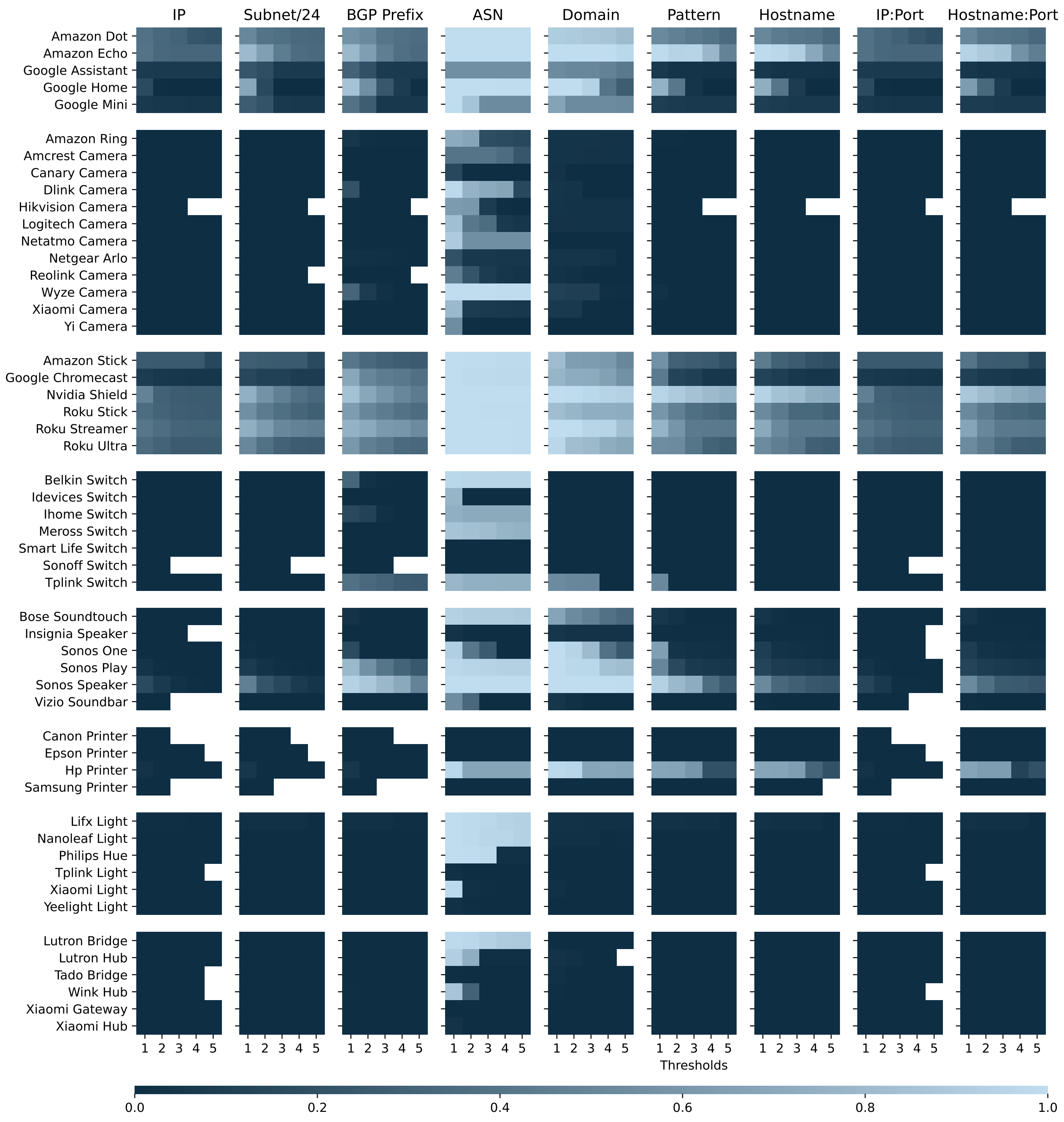


Figure 7.5: MFAF of allowlists using different host representations (columns). Rows represent the source product for generating the allowlists, and the color indicates the proportion of flows allowed when creating an allowlist using a threshold of between 1 and 5s (sub-columns) and applying it to devices of **the same type**.