THESIS

PROACTIVE EXTRACTION OF IOT DEVICE CAPABILITIES FOR SECURITY APPLICATIONS

Submitted by

Andrew Dolan

Department of Computer Science

In partial fulfillment of the requirements

For the Degree of Master of Science

Colorado State University

Fort Collins, Colorado

Spring 2020

ABSTRACT

PROACTIVE EXTRACTION OF IOT DEVICE CAPABILITIES FOR SECURITY APPLICATIONS

Internet of Things (IoT) device adoption is on the rise. Such devices are mostly self-operated and require minimum user interventions. This is achieved by abstracting away their design complexities and functionalities from users. However, this abstraction significantly limits a user's insights on evaluating the true *capabilities* (i.e., what actions a device can perform) of a device and hence, its potential security and privacy threats. Most existing works evaluate the security of those devices by analyzing the environment data (e.g., network traffic, sensor data, etc.). However, such approaches entail collecting data from encrypted traffic, relying on the quality of the collected data for their accuracy, and facing difficulties in preserving both utility and privacy of the data.

We overcome the above-mentioned challenges and propose a proactive approach to extract IoT device capabilities from their informational specifications to verify their potential threats, even before a device is installed. More specifically, we first introduce a model for device capabilities in the context of IoT. Second, we devise a technique to parse the vendor-provided materials of IoT devices and enumerate device capabilities from them. Finally, we apply the obtained capability model and extraction technique in a proactive access control model to demonstrate the applicability of our proposed solution. We evaluate our capability extraction approach in terms of its efficiency and enumeration accuracy on devices from three different vendors.

ACKNOWLEDGEMENTS

This work was funded in part by NIST under Contract Number 60NANB18D204, by funds from NSF under Award Number CNS 1650573, CNS 1822118, and funds from AFRL, SecureNok, Furuno Electric Company, and CableLabs. We would also like to thank Upakar Paudel for help with code development and experimentation. A special thanks to the Security Technologies team at CableLabs for all of their support throughout the course of this project.

DEDICATION

For Natalie, whose support and patience has been the incredible and constant wind in my sails.
For my family and my friends, without whom I would not have been able to produce what follows below.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

The popularity of IoT devices is gaining momentum, with projections of 75.44 billion devices worldwide by 2025 [1]. This large ecosystem is comprised of a variety of devices that are being used in diverse environments including healthcare, industrial control and automation, and smart homes. Manufacturers place emphasis on certain features or characteristics of IoT devices and often abstract away their actual design complexity and functionalities from the user. Many IoT devices are equipped with sensors and actuators beyond those required for their primary functionalities. For example, the Nest Protect [2], a smart smoke and carbon monoxide detector with an array of smoke and steam sensors as well as a microphone is able to perform far more than just smoke detection.

1.1 Motivation

These abstractions and extended (and in many cases hidden) functionalities of an IoT device result in a blind spot for consumers and leave an IoT ecosystem vulnerable to various security and privacy threats. Installing the previously mentioned smoke detector necessitates that the consumer understands its potential security and privacy consequences, especially considering the different locations that a smoke detector may be placed within a home setting. This requires the consumer to study the device's design specifications to find out what transducers it possess. Furthermore, she must have the insights to realize the security and privacy consequences of having a microphone that may be able to arbitrarily capture sound within the device's environment, and determine if any of those consequences violate the security policies of the household or organization. Performing all these steps is infeasible for most IoT users due to either their time constraints or lack of knowledge. Therefore, IoT consumers need assistance to properly interpret the underlying security and privacy threats from these devices. Our work aims to fill this gap by providing consumers information on IoT device capabilities and introducing ways that this information can be leveraged for the purposes of user-defined access control.

A comprehensive knowledge of device capabilities can be used in various security applications, including security verification, monitoring, risk analysis, and digital forensics. One example application is proactively verifying the security and privacy of IoT devices in a smart home or in an organization. Specifically, once the capabilities of a device are known, assertions can be made on whether or not those capabilities violate any of the security and privacy policies in an organization or a household. We can also ensure that the deployment of an IoT device in some location or under some configuration does not cause any security or privacy violations and can take adaptive measures to mitigate that risk.

While many other consumer goods and services are subject to large amounts of scrutiny when being considered for purchase, the same is not always true for IoT devices. Consumers are not expected to thoroughly understand the technologies that go into smart devices beyond the features as they are advertised. Despite the casual acceptance of smart devices and the abstractions of their inner workings, consumers often keep devices in deeply personal locations or interact with them in personal ways. The devices in these scenarios therefore come with security and privacy considerations that consumers should be aware of when using them, if not when purchasing them.

For example, the Nest Protect smoke alarm system provides users with the ability to receive notifications about potentially dangerous situations in their home. However, this device has an onboard microphone and is therefore capable of detecting (and recording) audio, which users may find concerning, depending on the physical location of that device in the home. However, the microphone is not advertised directly as part of any feature of the device, and is only listed within technical specification documentation for the device. An additional possibility is for a consumer to purchase this particular device for its primary features without being aware of the extended features of the device, including the potential for it to record sound.

Security goals and resulting policies often come more naturally to consumers than one might initially believe. Context of what a device is actually doing is important to these security goals, especially when put into simple terms that are intuitive for consumers to consider. He *et al.* conduct a survey in [3] to explore how users perceive different actions of IoT devices under access control circumstances related to social relationships, device location, and more. Device capabilities in this survey are not described in terms of actual device instances or particular manufacturers, but rather as abstract actions that could apply to many different devices. For example, "live video," "play music," and "camera on/off" are all capabilities described in the survey [3]. When these capabilities are posed to participants in conjunction with environmental attributes such as device location or the user who invokes the capability, participants present clear indications of security goals, such as the fact that device location is highly influential in access control decisions related to a device that performs live video.

The results of this survey are a clear indication that, when consumers are faced with device capabilities in plain terms, they can (and may even be more motivated to) make informed decisions about which device actions should be performed, who they should be performed by, and in what circumstances (based on social relationships, physical device location, etc.) they should be performed. In this way, knowledge and a clear understanding of device capabilities can empower users to make the security and privacy decisions that are best suited for their IoT ecosystem.

1.2 Problem Statement

This thesis primarily addresses the question of how to represent, extract, and leverage device capabilities for the ultimate benefit of user security and mitigation of privacy concerns. We aim to do so proactively, such that our proposed method could be used to garner insights about an IoT device and verify that it conforms to user-defined security policies even before it is installed in its environment. To this end, this thesis investigates the following questions:

- How can IoT device capabilities be defined and represented generically, across devices?
- How can such IoT device capabilities be extracted automatically from vendor-provided documentation materials for devices from different vendors?

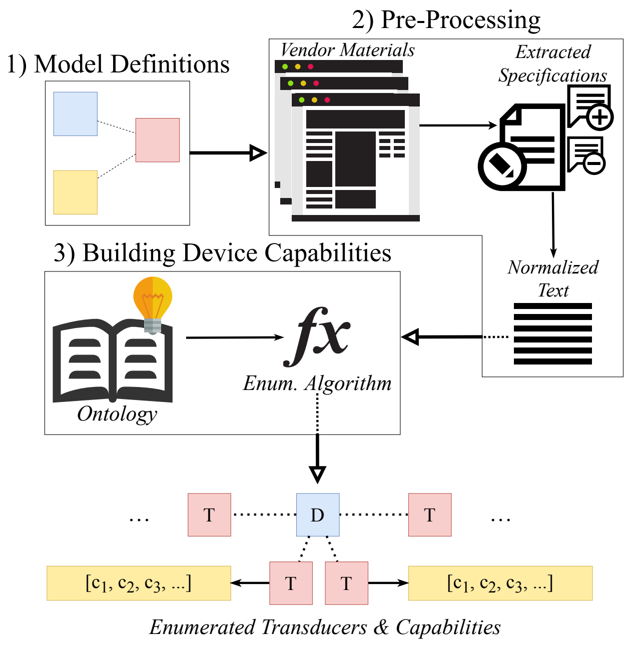


Figure 1.1: An overview of our methodology, including 1) development of the model, 2) pre-processing of vendor materials, and 3) applying an ontology to extracted specifications by way of an automatic enumeration function.

- How can these extracted capabilities be leveraged to verify the safety and security of an IoT device, relative to its environment?

1.3 Overview of Approach

Figure 1.1 illustrates an overview of our approach to extract the capabilities of a smart home device from its specifications. The three steps are described below, and we provide examples of various Google Nest products including the Nest Protect and Nest Cam Indoor [2, 4] throughout.

- **Step 1: Defining Capability Model for IoT Devices** We first define IoT devices, then define their transducers (i.e., sensors and actuators), and finally define the capability models that map transducers to their set of capabilities. (See Chapter 3).

- **Step 2: Normalizing Vendor Materials for Extraction** We first extract the device specifications from various vendor materials, then prune the extracted data to eliminate irrelevant contents (e.g., stop words and site navigation links), and finally normalize the pruned contents to refer to the transducer information. (See Section 4.1).

- **Step 3: Building IoT Device Capabilities** We first build an ontology of the device specifications (see Section 4.2), then derive an enumeration of transducers for a device by applying this ontology on the processed vendor materials, and finally map these transducers to their capabilities. (See Section 4.3).

1.4 Contributions

Several works [5–13] that profile IoT devices and their behaviors to detect security breaches and/or monitor an IoT environment pose two limitations: (i) Collecting and interpreting data from an IoT system is extremely challenging. Existing solutions [7, 8] either perform entropy analysis of encrypted traffic or use only the unencrypted features of network traffic (e.g., TCP headers and flow metadata). Due to its great reliance on data inference, false positives/negatives are a legitimate concern. Providing better accuracy in these security solutions is a critical challenge. (ii) Such approaches may reveal sensitive information (e.g., daily routines of smart home users [13]) about an IoT system and its users, threatening their privacy. Preserving privacy while sharing sensitive data for security analysis is another challenge.

We overcome these limitations and propose an approach to proactively extract IoT device capabilities from their vendor materials. Our work aims to provide a method for extracting IoT capability information in unambiguous terms. Additionally, we propose how these capabilities can be leveraged in an access control scheme that can provide users with fine-grained control over policies that match these intuitive and innate IoT security goals. We first define the notion of device capability in the context of IoT. Second, we extract the transducer (i.e., sensor and actuator) information for each device using vendor-provided specification and marketing materials. Third, we identify the capabilities of a device using an ontology that we derive that encodes capabilities of each sensor and actuator type. We discuss our approach in the context of smart homes, an important IoT domain with projections of 505 million active smart home devices worldwide by this year [14] and evaluate its efficiency and accuracy. Finally, we discuss the use of the capability information we extract for the application of fine-grained access control.

The main contributions of this thesis are as follows.

- We propose a new approach to proactively extract the device capabilities from design specifications. The key advantages of this approach over existing works are:

  (i) This approach does not rely on environment data and is therefore not directly affected by the difficulties of collecting and interpreting IoT data, and further is free from the privacy concerns of data sharing; and

  (ii) This approach enables proactive security verification of an IoT device even before it is installed or deployed.

- We are the first to define this concept of device capability in IoT, which can potentially be applied in the future security solutions for various IoT applications (e.g., smart grid, autonomous vehicle, smart health, etc.) to offer proactive security guarantee.

- As a proof of concept, we apply our extraction approach in the context of smart homes. We demonstrate the applicability of our approach by applying it to devices from various vendors (e.g., Google, Ring, and Alro), and we evaluate it in terms of its efficiency and accuracy.

- We introduce a scheme for fine-grained, user-defined access control in an IoT ecosystem that leverages the capability information that our extraction approach can provide.

1.5 Outline

The remainder of this thesis is organized as follows. Chapter 2 provides a background on the vendor materials that are analyzed in this work as well as our assumptions and threat model. Chapter 3 introduces our novel model of IoT device capabilities. Chapter 4 describes our methodology for extracting capabilities from vendor materials. Chapter 5 provides details on the implementation of our capability extraction methodology. Chapter 6 presents an evaluation of the implementation's performance in terms of efficiency and accuracy. Chapter 7 presents discussions on different aspects our implemented methodology. Chapter 8 discusses how capabilities extracted through our methodology can be leveraged in a fine-grained, user-defined access control scheme. Chapter 9 discusses related work and its limitations. Finally, Chapter 10 concludes this thesis and discusses future avenues for this work.

Chapter 2

Background

This chapter first provides a background on the vendor materials for IoT devices and then identifies the challenges to extract the capabilities from those materials; these challenges will later be addressed by our solution to extract capabilities from those vendor materials.

2.1 Vendor Materials

This section describes various vendor materials that provide the specification information of IoT devices.

2.1.1 Vendor Material Description

We consider vendor materials including product webpages, technical specifications, and developer documentations which are publicly available online and contain specification details of a device such as sensors, actuators, or related features.

Product Webpages. Product webpages are official marketing pages from which a consumer can purchase an IoT product, and contain the summary information about the device. For instance, Google has a page for each of its major Nest products (e.g., [15]). These pages can serve as an initial and often cursory source of information about a smart home device and its specifications.

Technical Specifications. Technical specification pages provide details about a device in terms of its hardware specifications. For instance, a technical specification page is available for each major Google Nest product (e.g., [4]). This work considers technical specification pages as the most significant sources of information about a device's hardware components, as they most often provide explicit enumerations of device components.

Developer Documentations. Developer documentations provide information to developers who create applications for or in conjunction with smart home devices. These materials usually include standardized software interfaces accessible as RESTful services, or software packages that can be integrated into other applications. Some vendors also offer entire platforms that can act as an abstraction on which different products can interoperate. Even though these materials are intended for application developers, they can be used as a source for the extraction of information about the hardware and capabilities of a device.

2.1.2 Investigation on Real-World Vendor Materials

We analyze the contents of several vendor materials for seven different devices by leveraging simple natural language processing techniques. These analyses result in insights on the challenges that come with the extraction of capability information from vendor materials, which is illustrated through the following examples.

Figure 2.1 shows the term frequency distribution for the Google Nest Cam Indoor's vendor materials as a word cloud, where the larger terms appear more frequently across the corpus of materials. This particular corpus is constructed from the Nest Cam Indoor main product page, technical specifications page, technical specifications support page, and the Nest Cam Developer API documentation [4, 15–17]. The full corpus contains 4,175 words after pre-processing. The word cloud suggests that terms that would intuitively be assumed to appear frequently, such as "camera" and "nest" appear often, as these terms are directly related to the primary functionality of the device. However, terms that are indicative of other transducers and their capabilities appear less often, and even appear less often than terms that are unrelated or potentially indicative of transducers that the device does not have. Figure 2.2 illustrates the frequency distribution of only a subset of notable terms.

From this distribution, it can be seen that the term "microphone" appears less often than "temperature," despite the fact that the Nest Cam Indoor has no transducer related to temperature. This particular disparity has to do with the "operation" and "storage" sections of the technical specifications page for the device, which can be seen in Figure 2.3. Other terms that are not related to any

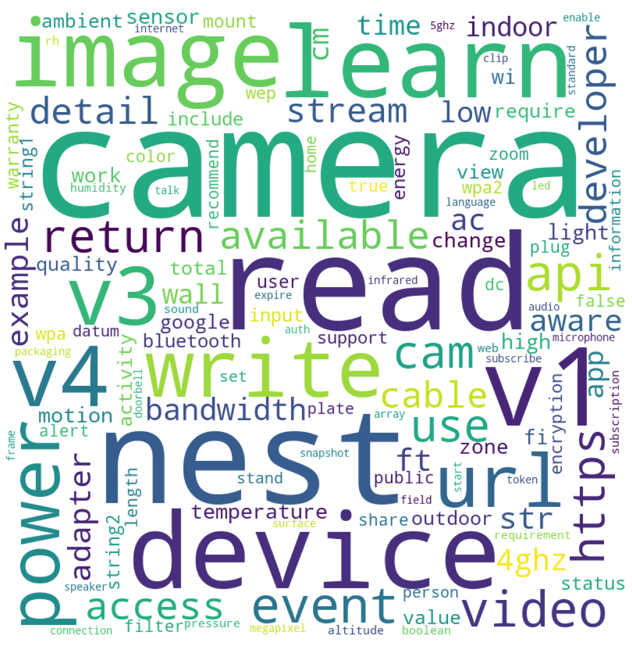


Figure 2.1: The term frequencies for the corpus of vendor materials on the Nest Cam Indoor, visualized as a word cloud.

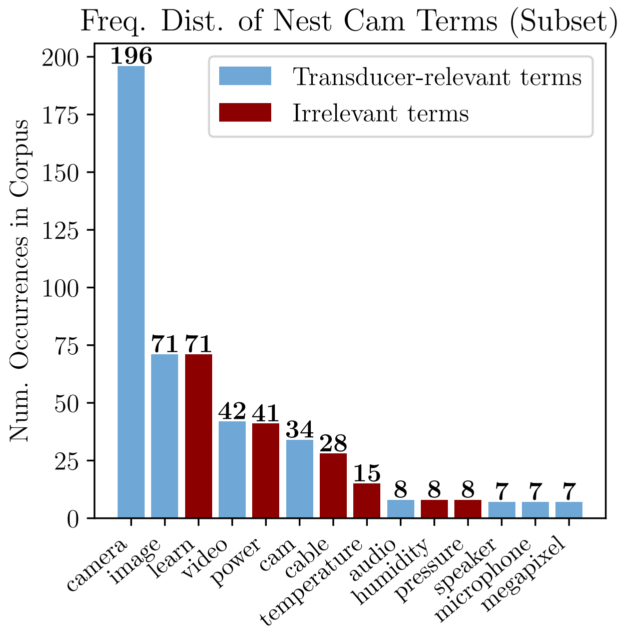
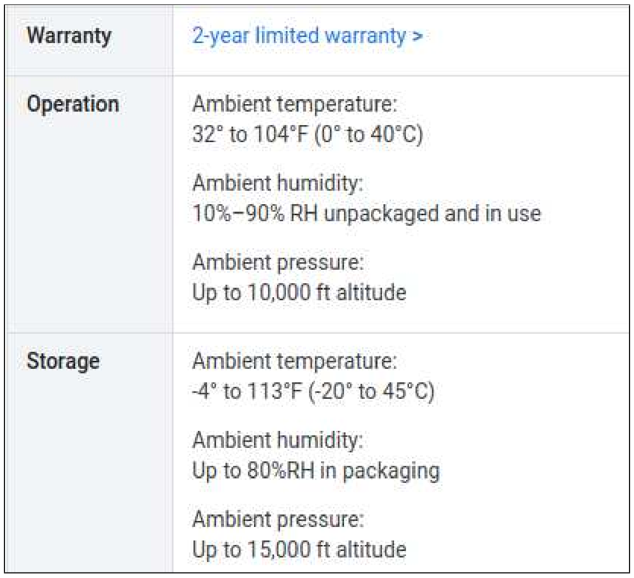


Figure 2.2: The term frequencies for a subset of terms from vendor materials on the Nest Cam Indoor. Terms that are more directly related to the transducer they refer to appear in blue, while other terms appear in red.



Warranty	2-year limited warranty >
Operation	Ambient temperature: 32° to 104°F (0° to 40°C) Ambient humidity: 10%–90% RH unpackaged and in use Ambient pressure: Up to 10,000 ft altitude
Storage	Ambient temperature: -4° to 113°F (-20° to 45°C) Ambient humidity: Up to 80%RH in packaging Ambient pressure: Up to 15,000 ft altitude

Figure 2.3: Different sections of the Nest Cam Indoor technical specifications page [16], where the terms "temperature" and "humidity" can be seen multiple times, but only in reference to the theoretical temperature range in which the device can regularly operate and be stored.

transducer such as "power" appear far more often (41 times) than terms of greater relevance such as "microphone," "speaker," or "audio" (each 7 times).

Additionally, term frequency-inverse document frequency (TF-IDF) calculations are also performed, treating each device's corpus as an individual document, with the goal of determining which unique words rank highly within each device's collection of vendor materials and therefore could be designated as most "relevant" to each device. An example of the results from this analysis are visualized for two devices in Figure 2.4, which displays the TF-IDF scores for a number of terms as well as the corresponding TF-IDF rank of each term within the vendor material collection for the Nest Cam Indoor and the Arlo Ultra [18] smart cameras. Note that the TF-IDF scores that are visualized in Figure 2.4 were computed using vendor materials for all seven devices that are evaluated in this work, not only the vendor material collections of the devices depicted.

Transducers that are directly related to the primary functionality of a device are associated with terms that tend to rank highly within the collection of that device's vendor materials in terms of TF-IDF. "Camera," for example, is among the top-ranking terms for both the Nest Cam Indoor and the Alro Ultra. Similarly, "smoke" and "alarm" both fall within the top 5 terms for the Nest

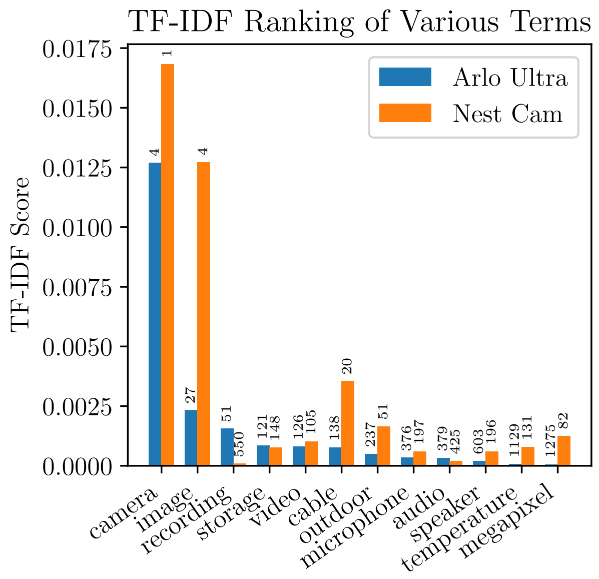


Figure 2.4: TF-IDF value and term rank within vendor material collections of the Arlo Ultra and Nest Cam Indoor smart cameras, computed using all device corpora by treating each device's vendor material corpus as an individual document.

Protect's vendor materials corpus. However, terms that are associated with other transducers that are not directly related to a device's primary functionality do not rank as highly. For both the Nest Cam Indoor and the Arlo Ultra, the term "microphone" ranks 197th and 376th within each respective device corpus, likely due to its low frequency of appearance within each corpus. The same can be said for the corpus of the Nest Protect, where "microphone" ranks 270th among all unique words.

Overall, TF-IDF fails to highlight many terms that are most indicative of a device's transducers and capabilities due to their infrequent appearances, and therefore is not a sufficient method to discover which transducers and capabilities are present on a device. While TF-IDF is well suited for keyword extraction given an extensive corpus of written articles, there are a number of challenging aspects of IoT device vendor materials that limit its use for our purposes.

2.1.3 Challenges in Extracting Capabilities from Vendor Materials

Based on the outcome of the analysis above, we enumerate the major challenges in extracting capabilities from vendor materials as follows.

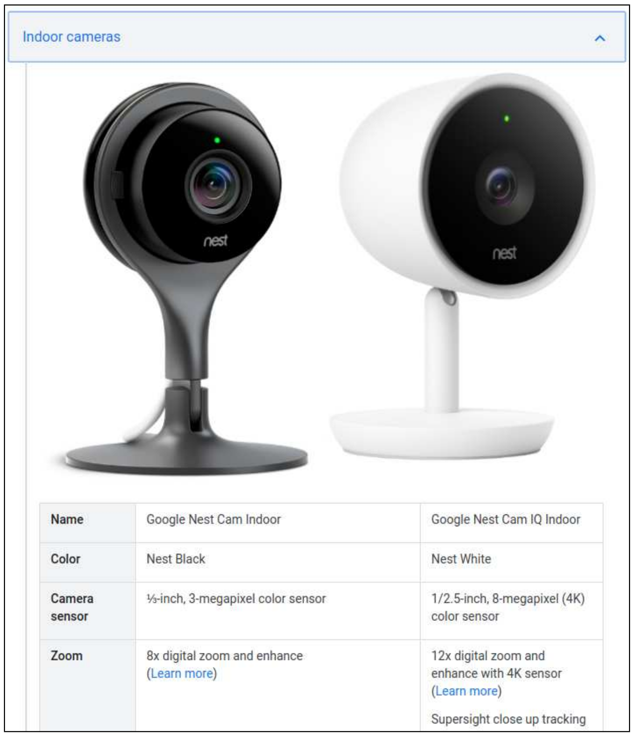
No Standardized Template. Each vendor follows different templates for their materials and furthermore, different materials of the same vendor follow different formats. There is no standard template or specification for how to describe different generic features or hardware components of IoT devices. This implies significant effort to extensively learn those different templates to extract information from them comprehensively.

Brevity of the Materials. As vendor materials are usually meant for a consumer audience, they are expressed in a brief manner and do not include all explicit specifications of a device. Therefore, extracting device information from them requires more interpretation of their contents. Additionally, terms that are indicative of particular hardware components may only appear a limited number of times within the materials, especially if they are not related to the primary function of the device.

Vendor-Specific Jargon. Each vendor tends to use their own set of terminologies when describing their devices. As the primary purpose of vendor materials is to inform potential purchasers about a device, vendors tend to craft language around what information they believe is the most useful or well-received in the eyes of the consumer, and include terminology that may be unique to only their line of products. For example, one feature of the Nest Protect is advertised on the overview page as "Steam Check." Accordingly, learning the vocabularies used for various vendors and then normalizing them to infer their capabilities presents additional challenges.

Interpreting Visual Contents. Several contexts of vendor materials are represented visually and are therefore difficult to detect automatically. An interesting aspect of the marketing and technical specification pages for IoT devices is the way that page layout and structure provide contextual information in the form of visual cues and hierarchical organization. In these cases, text processing alone becomes insufficient for comprehensive extraction of transducer information. For example, Figure 2.5 displays an example layout of technical specifications for multiple Google Nest cameras [16]. Though this figure only shows a small portion of the page, it is clear that there is a great amount of information carried in its layout.

Distribution of Information. Information about any one device is distributed over various materials. For the most thorough extraction of transducer and capability information, it is essential that


Indoor cameras

Name	Google Nest Cam Indoor	Google Nest Cam IQ Indoor
Color	Nest Black	Nest White
Camera sensor	⅓-inch, 3-megapixel color sensor	1/2.5-inch, 8-megapixel (4K) color sensor
Zoom	8x digital zoom and enhance (Learn more)	12x digital zoom and enhance with 4K sensor (Learn more) Supersight close up tracking

Figure 2.5: Use of page layout to convey information about different Nest camera devices [16]. Note that the two columns of the table are labeled primarily by the images of each device.

as many different materials as can be found are utilized. However, different materials may or may not be available, and some may be less accessible than others.

2.2 Threat Model

We focus on smart homes, a prominent domain of IoT, in this work. We assume that the sensors and actuators in a smart home device may be used to conduct various security and privacy attacks. Our approach, therefore, builds the device capabilities (i.e., the actions that a device can perform) which can later be used to detect/prevent the adversaries that exploit these sensors or actuators. Our approach does not consider the threats related to a malicious or vulnerable transducer, which includes misbehavior and malfunction. Also, any network attack that does not involve the transducers is beyond the scope of this thesis. In this work, we derive the device capabilities from the vendor-provided materials that are publicly available. Therefore, any missing information about a device in those materials may affect the effectiveness of our approach.

Chapter 3

A Capability Model for IoT Devices

The primary goal of this work is the automatic extraction and enumeration of device capabilities from vendor-provided documentation materials. We present in this chapter a model that captures device capabilities, as well as the transducers to which they are associated, as generics. This representation allows us to consider transducers and their associated capabilities without any dependencies on their actual implementations in hardware or software.

3.1 Definitions

This work defines a *transducer* as a sensor or actuator (inspired in part by the definitions in NIST 8228 [19]). A *sensor* holds the core functionality of sensing or measuring various aspects of a physical environment and converting such measurements to a digital signal. For example, image sensors, motion sensors, and microphones sense light, motion, and sound from a physical environment, respectively. An *actuator* converts a digital signal to some physical action(s), such as emitting light, producing sound, or actuating a lock to toggle its state. Denoted formally, the set of all transducers T is partitioned into the set of all sensors S and the set of all actuators A, where $T = S \cup A$ and $S \cap A = \emptyset$ (no sensor is also an actuator).

A device *capability* is a function that a device is able to perform. This work makes the assumption that each transducer has associated with it a static set of capabilities that are intrinsically bound to that transducer. In the case of the Nest Protect, a microphone is innately capable of capturing audio, while a smoke sensor is innately capable of detecting smoke. A capability of a sensor is denoted as c_{si} and a capability of an actuator is denoted as c_{aj}. Note that, $\forall i, j, c_{si} \neq c_{aj}$. However, for any two sensors s_m and s_n, or any two actuators a_m and a_n, where $m \neq n$ $s_m, s_n \in S$, and $a_m, a_n \in A$, the set of capabilities may overlap.

A transducer t_i is either a sensor s_i or an actuator a_i. Each sensor s_i and each actuator a_i are represented by a non-zero finite set of capabilities, denoted as $s_i = \{c_{s1}, c_{s2}, \ldots, c_{sp}\}$ and $a_i = \{c_{a1}, c_{a2}, \ldots, c_{aq}\}$. Additionally note that $s_i \neq \{\}$ and $a_i \neq \{\}$.

An IoT *device* is defined as an embedded system that consists of a set of transducers. A device D_i consists of a set of sensors S_i and actuators A_i where $S_i \subseteq S$ and $A_i \subseteq A$ and $S_i = \{s_1, s_2, \ldots, s_n\}$ and $A_i = \{a_1, a_2, \ldots a_m\}$. The total number of transducers in D_i is therefore $n + m$.

As an example, the Nest Protect and a subset of its transducers could be represented in our model as a device D_1 with the following transducers: a smoke sensor s_1, a microphone s_2, a motion sensor s_3, and a speaker a_1. The device D would therefore be represented as $D_1 = S_1 \cup A_1$, where $S_1 = \{s_1, s_2, s_3\}$ and $A_1 = \{a_1\}$. Assume the capabilities of the smoke sensor s_1 to be smoke detection c_1. Assume the capabilities of the microphone s_2, motion sensor s_3, and speaker a_1 to be audio capture c_2, motion detection c_3, and produce audio c_4, respectively. The full set of capabilities this representation of the Nest Protect D_1 would then be $\{c_1, c_2, c_3, c_4\}$.

3.2 Model Relationships

Transducers are represented in this work independent of their implementation details and focuses on their generic types, instead of transducer instances. As a result, devices and transducers have a many-to-many relationship, where a particular transducer type can be associated with many different devices, and devices can have a variety of transducers on board.

Consequently, multiple devices may have common sensors or actuators. For example, both the Nest Protect and the Nest Cam Indoor have microphones. Two devices D_r and D_s shown below have common sensor s_3 and common actuator a_2.

$$D_r = S_1 \cup A_1 = \{s_1, s_2, s_3, a_1, a_2\}; \; D_s = S_2 \cup A_2 = \{s_3, s_4, a_2, a_3\}$$

The set of capabilities for a device D_i is computed as the union of the set of capabilities of the transducers comprising the device. Multiple devices can share common capabilities by either having a common generic transducer or by having multiple transducers that share a common capability.

For example, the Nest Cam Indoor has an image sensor and therefore holds the capabilities of image capture and light detection. On the other hand, the Nest Protect has an ambient light sensor, which also holds the capability of light detection. This capability is shared between the two devices, despite the fact that the transducers that the capability corresponds to are different. In the formalization below, the two devices D_p and D_q have common capabilities c_{s3} and c_{a2}:

$$D_p = \{c_{s1}, c_{s2}, c_{s3}, c_{a2}, c_{a3}\};\ D_q = \{c_{s3}, c_{s4}, c_{s5}, c_{a1}, c_{a2}\}$$

Chapter 4

Extracting Capabilities from Vendor Materials

4.1 Normalizing Vendor Materials for Extraction

To prepare the vendor materials for building device capabilities, we first extract the device specifications from the vendor materials, then remove irrelevant information (e.g., external navigation links or copyright information) from those extracted specifications, and finally refine them into a more homogeneous, and machine-friendly format.

4.1.1 Selective Extractions of Device Specifications

The initial extraction of the device specification is a process that operates on the input vendor materials. The vendor materials must be parsed for their contents, which can be defined in terms of semantics as well as more abstract information such as document structures and page layouts. In our work, we assume the vendor materials are presented as HTML or text documents.

For HTML documents, we first remove the non-HTML contents from each web page, such as style and script blocks. We then extract the raw text from the resulting HTML elements. The information that is most pertinent to this work often only makes up a subset of a web page's contents. Other elements of the page including navigation links (to other products, for example) can introduce terms that are unnecessary or that could even be misleading from the perspective of an automated extraction system concerned with the device that is represented by the page. To overcome this challenge, we parameterize this step so that specific sections of a page can be extracted. For example, the technical specifications page for the Nest Protect contains page elements unrelated to the device's hardware or functionalities, including navigation links to other products and shipping information. With our parameterized method, we are able to extract only the specification information for the device. We also tailor the parameters to specific vendor pages; these parameters are often reusable, as web design within a single vendor is often homogeneous.

4.1.2 Normalizing Material Content

To ensure that the contents extracted from the vendor materials are best suited for our transducer enumeration technique, our approach normalizes those contents by removing elements that are not critical to the extraction process, including punctuation, non-alphanumeric, and non-white-space characters, as well as "stop words" (i.e., common articles and prepositions in English). The case and plurality of the resulting terms are also normalized through lemmatization, a process of linguistics that simply involves homogenizing different inflections of the same term to its dictionary form. The content output by the normalization step contains a more homogeneous sequence of terms in the original order that they appeared in the vendor materials.

For example, the Nest Protect's overview page contains the following text: *"In addition to its voice, Nest Protect uses colors to communicate. And we gave the new Nest Protect a brighter light ring so it's easier to see in an emergency and better at helping you see things in the dark"*. After the normalization step, the processed string will read *"addition voice nest protect use color communicate new nest protect bright light ring easy emergency good help thing dark"*.

4.2 Building an Ontology

The inferences that must be taken into account to successfully interpret what transducers and capabilities a device has often come with prerequisite technical knowledge that end consumers are not expected to have. The same goes for a system that performs this interpretation automatically, where an understanding of the language and structure of vendor materials must be encoded into the process itself for the most accurate results. In other words, an ontology must be created and leveraged as the primary source of knowledge behind the enumeration of transducers and the mapping to capabilities.

In this work, we develop an ontology that aims to provide an understanding of the terminologies used to refer to specific components of a device, whether they are explicit, ambiguous, or uniquely created and defined by the vendor. Additionally, we attempt to capture the general relationships between abstract device types and their common transducers; for example, the abstract device type "smoke alarm" commonly has smoke sensor, carbon monoxide sensor, motion sensor, microphone, temperature sensor, humidity sensor, light sensor, and speaker transducers.

It is worth noting that other characteristics of vendor materials could be encoded and utilized for capability extraction. For example, because images of devices appear on product web pages, an ontology could be created with an encoding of image-based knowledge that could equip the system with an awareness of visual aspects of a device that provide hints of the presence of certain transducers. An ideal system would be able to understand vendor materials to a capacity that is as good or better than that of humans, including the ability to recognize transducers visually. Realistically, an ontology of this nature can never be fully complete. An ontology can be continuously augmented with new information to improve transducer enumeration; this new knowledge can be found in a number of ways, including manual review of vendor materials or more automated solutions that are outside the scope of this work. We create the initial ontology by manually reviewing vendor materials in detail to capture the aspects described above.

4.2.1 Review of Vendor materials

The goal of the manual review step is to enumerate the different hardware components and capabilities of a device for a baseline of ground truth, and also to enumerate and analyze the inferences and assumptions that are required for the reader to draw these conclusions. These results are captured and become the basis for the ontology that will be used as a parameter for the automated enumeration process.

The process of manually reviewing vendor materials is the same for all devices. This process involves reading through different documents that are associated with each device and interpreting from them the set of transducers (and capabilities) of the device. During this process, any insights, inferences, or context clues that are used in this interpretation are also captured. The most important of these features are key terms that act as indicators of the presence of a particular transducer or capability. A *key term* is defined in this work as one or more words that refer to a particular concept.

Table 4.1: Obtained ontology from the Nest Cam Indoor materials

Category	Transducer	Capabilities	Example Terms
Sensors	Image Sensor	Capture Image, Capture Video, Detect Light	image sensor, camera, video
	Microphone	Capture Sound	microphone, voice, audio
Actuators	Speaker	Produce Sound	Speaker, Audio, Two way audio
	LED Light	Produce Light	LED light
	Infrared Light	Produce Infrared Light	Infrared, IR, Night vision

For example, when reviewing the Nest Cam Indoor's technical specifications on the Google Nest support forums [16], simple terms such as "camera" are indicative of the presence of an image sensor. On the other hand, the term "video" is more ambiguous, and could refer to the video captured by the image sensor, or video displayed on some kind of screen. In this case, it can be "inferred" that this term refers to an image sensor because of context clues provided by additional related terms such as "1080p," which refers to the resolution of the video captured by the sensor, and "lens," which refers to the lens of the camera. On their own, these additional terms do not necessarily suggest the presence of an image sensor, but they can provide context when considered in conjunction with other camera-related terms to suggest with higher confidence the presence of the sensor. The understanding of these terms as context clues carries an assumed level of prerequisite knowledge of camera-related terminology.

The visual cues provided by page structure can also help a reader understand the context around the terms. For the Nest family of products, it is common for the term "temperature" to appear on technical specification pages for devices that have no sensors or actuators related to the measurement or alteration of any temperature. Instead, these instances of the term are used to describe the "operating" constraints of the device (the theoretical range of temperature in which it can operate). The only real indicator of this difference is in the table layout of the Nest Cam Indoor's technical specifications page, where a reader can see by way of the row's label "operation" that the temperature in this case refers to the device's operating temperature and not any sensor. This part of the page is illustrated in Figure 2.3.

Perhaps the most abstract feature that is considered during manual review, which is largely dependent on the individual reviewer, is the use of technical background knowledge to infer, from a described concept, how a certain feature of a device may be implemented. The term "motion detection," for example, could indicate the presence of a motion sensor or of software that enables an image sensor to perform motion detection. Regardless, the ability for the reader to conceive of these possibilities is dependent on their technical knowledge.

A sample enumeration of transducers and their capabilities for the Google Nest Cam Indoor is summarized in Table 4.1. This table represents a portion of an ontology that is derived from the manual review process. The ontology contains this enumeration of transducers and their corresponding capabilities, as well as some key terms that refer to the transducers. The ontology also contains an understanding of which transducers are commonly part of different abstract device types. In the case of a general smart camera, the transducers enumerated in Table 4.1 are common. Note that our example of a motion-activated smart camera does not fit perfectly into the abstract device type of a smart camera, due to the presence of a motion sensor. An additional abstract device type could account for this extra sensor.

4.3 Extracting Device Capabilities

To enumerate device transducers using the above-mentioned ontology, we devise three algorithms, namely, device cognizant key term set matching (dcKTSM), indicative key term set matching (iKTSM), and all key term set matching (aKTSM), where a key term is a set of one or more words related to a transducer. We further consider two types of key terms: indicative terms and related terms. *Indicative terms* are considered to be unambiguously indicative of the presence of a transducer. *Related terms* are related to a transducer, but may be more ambiguous, and hence are not sufficient for drawing conclusions about its presence. For example, in the case of the Nest Protect, the term "microphone" is considered indicative of a microphone, while the terms "audio" or "voice" are considered related to a microphone.

The primary purpose of the proposed algorithms is to use different key terms found within a corpus of normalized vendor materials text to determine which transducers are present. The algorithms operate on a corpus of vendor materials that are assumed to be related to a single device. They additionally use the ontology described in Section 4.2 as input to apply encoded knowledge for the extraction process. For our purposes, the ontology contains information about terminologies used to describe devices, as well as details on the understanding of abstract device types.

Using these inputs, the algorithms produce enumerations of transducers that can then be mapped to their static set of capabilities (which are represented in the ontology). The transducers that are enumerated from the extraction step must be represented in a standardized way, (e.g., using the same identifier for the transducer type), where each extracted transducer is completely decoupled from the device instance it was extracted from. This is to ensure that the transducers remain generic and therefore fit into the model that is described in Chapter 3.

The main difference between the three algorithms is the broadness of their matching scope. Device cognizant KTSM can be considered the most constrained, as it only matches with indicative terms for a particular abstract device type. Indicative KTSM uses all indicative terms for matching, without considering abstract device type, but does not use any related terms. Finally, all-KTSM uses both indicative and related terms in the ontology, and therefore is most broad in matching scope. While a broader matching scope may result in more device transducers being identified, it may also introduce more incorrect matches of transducers from the ontology that are not actually on the device. A less broad matching scope results in the inverse, where less transducers are correctly identified and less incorrect matches of transducers that are not actually on the device.

For the example of the Nest Protect, consider the only indicative term of the microphone sensor to be "microphone." Assume the related terms for the microphone sensor include "voice," "sound," and "listen." It may be the case that the indicative term for this transducer never appears in the corpus of vendor materials. In that case, the algorithms that are less broad in matching scope could fail to enumerate that transducer. On the other hand, the other algorithms could enumerate the transducer, but may also introduce incorrect matches if the related terms are also associated with other transducers that are not actually on the device. We describe each algorithm below.

4.3.1 Device Cognizant Key Term Set Matching (dcKTSM)

The Device Cognizant Key Term Set Matching algorithm utilizes the ontology's knowledge of abstract device types to perform transducer enumerations. The ontology contains an understanding of abstract device types and the transducers that are common to them. This knowledge can then be leveraged in order to first estimate which abstract device type is represented by the corpus of the vendor materials. From there, the indicative terms for that abstract device type are used to determine which of its transducers are present.

Algorithm 1 shows the dcKTSM algorithm, which first filters the key term sets for different generic device types, and then performs key term matching based on the most relevant set of indicative terms. Specifically, Lines 1-11 outline the first matching step, which uses both indicative and related terms to determine which abstract device type is most likely being represented by the corpus of vendor materials. The indicative terms of the device type that is ranked as the best match candidate are considered to be the terms that refer to the transducers of the device. A reverse-mapping step is then performed on Line 12, where only these indicative terms are used to determine which transducers are present. Note that the output of the first matching step is a subset of indicative terms for the selected abstract device type. The reverse mapping step determines which transducer each indicative term refers to, where some transducers may be referred to by more than one indicative term. The final results contain an enumeration of transducer identifiers, which are returned at Line 12.

The dcKTSM algorithm is able to better exploit context clues found in the related terms while avoiding erroneous matches that can be introduced by their ambiguity. Additionally, the approach accounts for the fact that terms that are most directly indicative of the presence of a transducer may have a frequency that is much lower than that of other terms. For example, even if the indicative term "microphone" appears only twice within an entire corpus, the presence of related terms such as "audio" or "voice" can help bolster confidence when concluding that a microphone transducer is present.

Algorithm 1: Device Cognizant Key Term Set Matching (dcKTSM)

```
1  best_score ← 0
2  best_matches ← **null**
3  for d in Ontology[Devices] do
4      indicative_terms ← d[transducers][indicative_terms]
5      related_terms ← d[transducers][related_terms]
6      ind_matches ← get_matches(indicative_terms, corpus)
7      rel_matches ← get_matches(related_terms, corpus)
8      match_score ← |rel_matches| + |ind_matches|
9      if match_score > best_score then
10         best_score ← match_score
11         best_indicative ← ind_matches
12 transducer_identifiers ← reverse_map(best_indicative)
```

4.3.2 Indicative and All Key Term Set Matching (iKTSM and aKTSM)

Algorithm 2 uses only the indicative terms without ranking term sets by potential abstract device type. Specifically, it evaluates the corpus for any matching indicative terms of any transducer, and identifies the transducers through the same reverse-mapping process (as in Algorithm 1).

Algorithm 2: Indicative Key Term Set Matching (iKTSM)

```
1  all_indicative_terms ← $\bigcup_{d}$ Ontology[Devices][d][transducers][indicative_terms]
2  ind_matches ← get_matches(all_indicative_terms, corpus)
3  transducer_identifiers ← reverse_map(ind_matches)
```

Additionally, we consider a completely unguided KTSM alrogithm called all-KTSM (aKTSM), shown in Algorithm 3, that performs the same matching as indicative KTSM, but also matches on the related terms. In other words, this algorithm matches on all terms that are defined in the ontology, regardless of whether they are related or indicative. Accordingly, we consider this algorithm to be the most broad in matching scope and therefore least exact.

Algorithm 3: All Key Term Set Matching

```
1 all_indicative_terms ← $\bigcup_{d}$ Ontology[Devices][d][transducers][indicative_terms]
2 all_related_terms ← $\bigcup_{d}$ Ontology[Devices][d][transducers][related_terms]
3 all_terms ← all_indicative_terms + all_related_terms
4 all_matches ← get_matches(all_terms, corpus)
5 transducer_identifiers ← reverse_map(all_matches)
```

Table 4.2: An excerpt of outputs from our approach.

Device	Category	Transducer	Capabilities
Arlo Ultra [18]	**Sensors**	Image Sensor	Capture image, Capture video, Detect light
		Microphone	Capture sound
		Motion Sensor	Detect motion
	Actuators	Speaker	Produce sound
		LED Light	Produce light
		Infrared Light	Produce IR light (enabling night vision)
		Siren	Produce high-volume siren
Nest Cam Indoor [4]	**Sensors**	Image Sensor	Capture image (take photo), Capture video, Detect light
		Microphone	Capture sound
	Actuators	Speaker	Produce sound
		LED Light	Produce light
		Infrared Light	Produce IR light (enabling night vision)
Nest Protect 2nd Gen [2]	**Sensors**	Smoke Sensor	Detect smoke
		Carbon Monoxide Sensor	Detect carbon monoxide
		Temperature Sensor	Measure temperature
		Humidity Sensor	Measure humidity, detect steam
		Microphone	Capture sound
		Motion Sensor	Detect motion
		Light Sensor	Detect light
	Actuators	Speaker	Produce sound
		LED Light	Produce light
Nest X Yale Lock [20]	**Sensors**	Light sensor	Detect light
		Touch Sensor	Detect (capacitive) contact
	Actuators	Lock	Lock and unlock door
		Speaker	Produce sound
		LED Light	Produce light

4.3.3 Mapping to Device Capabilities

Capabilities of a device are enumerated from its constituent transducers. This work assumes that transducers are associated with a static, finite set of capabilities that are established during the creation of the ontology. Each transducer can be directly mapped to its set of capabilities as per Chapter 3. The capabilities contained in the final output set represent a device's functionality unambiguously as it is described by the vendor materials. Table 4.2 provides examples of the outputs of our extraction methodology, including this mapping step.

Chapter 5

Implementation

We build an ontology using the manual review process described in Section 4.2.1 from a variety of vendor materials for seven smart home products: Arlo Ultra Camera [18], Nest Cam Indoor [15], Nest Hello Doorbell [21], Nest Protect [2], Nest Learning Thermostat [22], Nest X Yale Lock [20], and Ring Indoor Camera [23].

5.1 Pre-processing and Normalization

The pre-processing routine, implemented in Python, can fetch the product web pages directly over the network via the requests library [24] by way of their URI, or read pages fetched previously and saved locally. Pages are typically saved locally to ensure consistency between experiments, and as a precaution against the contingency that web pages become unavailable.

To extract the textual content from those pages, we utilize the BeautifulSoup package [25], which allows us to extract only specific sections of vendor material pages by providing parameters with specific HTML tags and attributes used to identify page portions. Our current implementation supports the static (HTML) elements in web pages, which is the current format for most vendor materials. However, if some vendor materials only display their content dynamically, a simple workaround would be to use a web engine to first internally render any dynamic content before processing the resulting HTML. To normalize the text, a separate Python function replaces stop words and non-alphanumeric characters via regular expressions. For the normalization of term plurality, lemmatization is performed using the Spacy natural language processing package [26].

5.2 Extraction Algorithms

The KTSM algorithms described in Section 4.3 are also implemented as Python functions that operate on a corpus from which to enumerate transducers. To encode the ontology created during the manual review process (as described in Section 4.2.1), we use a JSON-based data model to represent abstract device types, transducers, their capabilities, and key term sets. Each transducer is represented in the data model as having a static set of capabilities, a set of indicative terms, and a set of related terms. Additionally, each transducer has a unique identifier. Abstract device types are represented in the data model as collections of these transducer identifiers, allowing our implementation to map from abstract device type to specific transducers and their term sets. These data models act as additional parameters to our KTSM functions.

Given the corpus of a device's vendor materials and the data model, the implemented KTSM functions utilize the tree-based flashtext algorithm [27] to perform efficient key term matching. This key term matching functionality does not support partial matches (as regular expressions do), so distinct key terms must be used. It is also possible for key terms to overlap; for example, "image sensor" and "image" are considered different key terms that have the same first token. For these cases, flashtext counts the longest match, and does not count such terms more than once. In the previous example, "image sensor" would be counted instead of "image."

In the case of device cognizant KTSM, matches are extracted for indicative and related terms. The total number of matches is first used to determine the best abstract device type, and the matching indicative terms of the best candidate device type's transducers are returned. In indicative and all-KTSM, the matching step takes place with all indicative terms, and with all indicative and related terms, respectively, and all resulting matches are returned. Matches are mapped to their associated transducer's unique identifier automatically, by way of a feature of the Python implementation of flashtext.

Chapter 6

Evaluation

This section discusses the performance of our implemented solution, which is used to extract enumerations of transducers for the seven devices.

6.1 Experimental Setting

All extractions are performed on a system with an Intel Core i7-8550U processor @ 1.80 GHz and 8 GB of memory. We evaluate the performance of our implementation in terms of extraction efficiency, accuracy of transducer enumeration, rate of incorrect transducer matches (false discovery rate), and the overall precision and recall of each algorithm.

6.2 Efficiency

The goal of the first set of experiments is to measure the efficiency of our approach. The efficiency of our implementation refers to the total time required to perform the pre-processing step on the input vendor materials for a particular device. Displayed in Figure 6.1, the extraction step is the most significant source of processing time in our methodology, ranging from less than a second to nearly 20 seconds.

The time required for the extraction step depends on the size of the vendor materials that are used as input. HTML files for devices may exhibit a large range of sizes; for example, pages on Google Nest devices have the largest range of sizes on disk, from 38 KB to 2.6 MB. The HTML extraction portion of the pre-processing step takes the largest amount of time, between 15 and 18 seconds, for the largest collections of web pages (over 4 MB total). Comparatively, smaller collections of pages (totalling 200 KB or less) take less than a second for extraction. Figure 6.1 illustrates that extraction time scales with collection size linearly.

For most evaluated devices, the times required to perform the text normalization step, displayed in Figure 6.2, are negligible when compared to those of the extraction step. Consistently, across

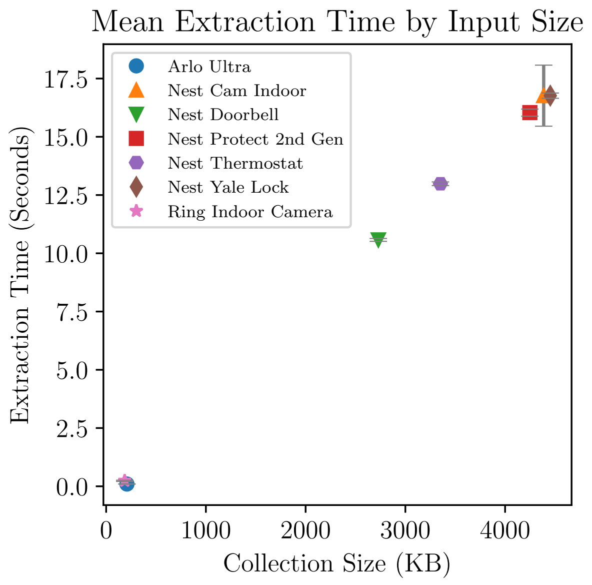


Figure 6.1: Average text extraction time of the vendor materials for different devices by their size on disk, computed over 5 trials.

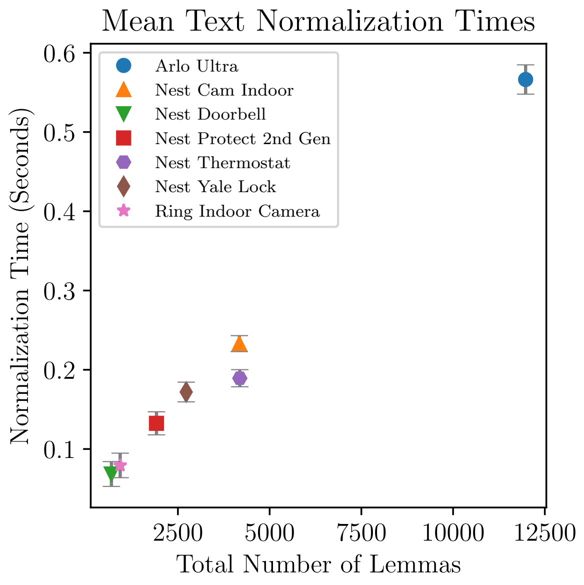


Figure 6.2: Average text normalization time by the total number of lemmas in a corpus, computed over 5 trials.

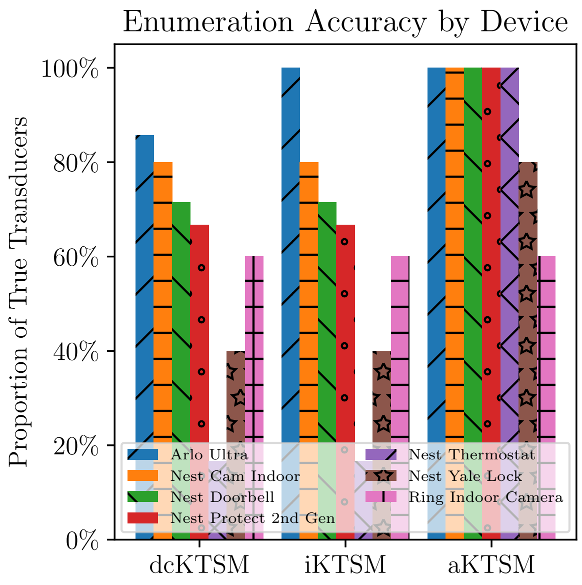


Figure 6.3: For each device, grouped by KTSM algorithm, the proportion of ground truth transducers that are correctly enumerated.

all devices, the text normalization step is performed in less than a second, increasing slightly with the total number of lemmas extracted.

The automatic enumeration of device transducers from normalized text has little impact on the efficiency of our solution, due to the efficiency of the tree-based flashtext algorithm for keyword extraction which we employ. We therefore don't report the amount of time that the keyword extraction takes. Additionally, the time required for mapping from enumerated transducers to their capabilities is not included in our efficiency measurements, due to the fact that this mapping is statically defined in the ontology. That is, once the transducers have been enumerated, establishing their corresponding capabilities requires a trivial lookup in the ontology (implemented as a Python dictionary, in our case).

6.3 Enumeration Accuracy

For the purposes of evaluation, a sample set of transducers for the evaluated devices are enumerated manually as ground truth. Enumeration accuracy is then computed as the true positive rate, the proportion of these ground truth transducers that were correctly identified by an extraction

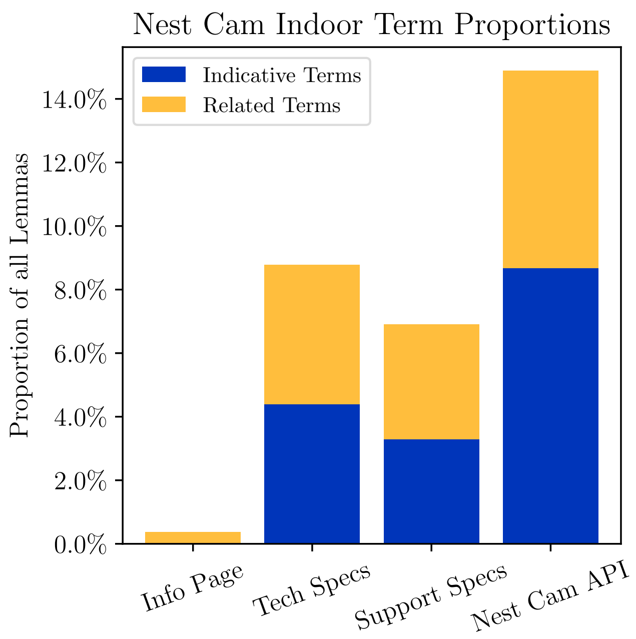


Figure 6.4: A comparison of the proportions of indicative and related terms per document for the Nest Cam Indoor [4].

approach. Figure 6.3 displays the results of our enumerations on the seven different devices from three different vendors, grouped by enumeration algorithm.

Figure 6.3 shows that both the device cognizant and indicative KTSM algorithms provide similar proportions of correctly identified transducers, averaging about 60% and 62%, respectively, of ground truth transducers identified among all devices with a standard deviation of 24% and 27%, respectively. In contrast, the all-KTSM approach averages an enumeration accuracy of 91%, with a standard deviation of 16%.

For each extraction approach, there is a large disparity between the transducer enumeration accuracy for the Nest Cam Indoor and the Nest X Yale Lock. We present a comparison of the composition of the vendor material corpora for these two devices in Figure 6.4 and Figure 6.5, showing the proportion of indicative and related terms for each document per device. Seen in these figures, while not all documents in the Nest Cam Indoor corpus consist of a large proportion of indicative terms, the entire corpus contains a higher proportion of indicative terms overall when compared to the Nest X Yale Lock Corpus.

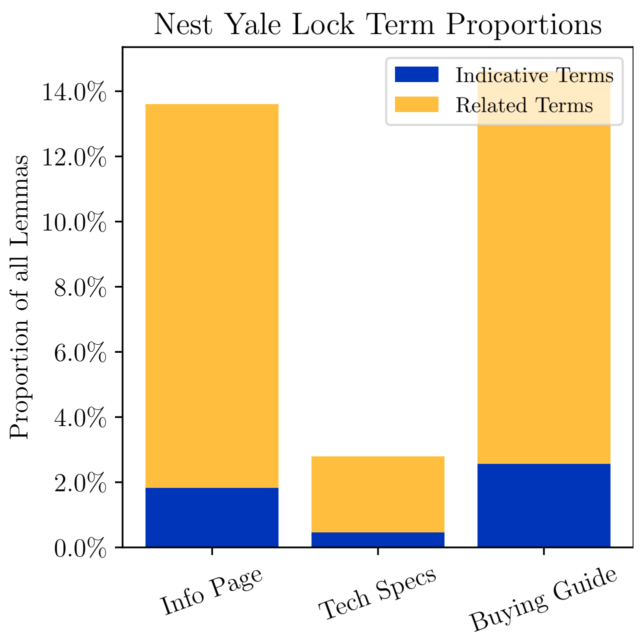


Figure 6.5: A comparison of the proportions of indicative and related terms per document for the Nest X Yale Lock [20].

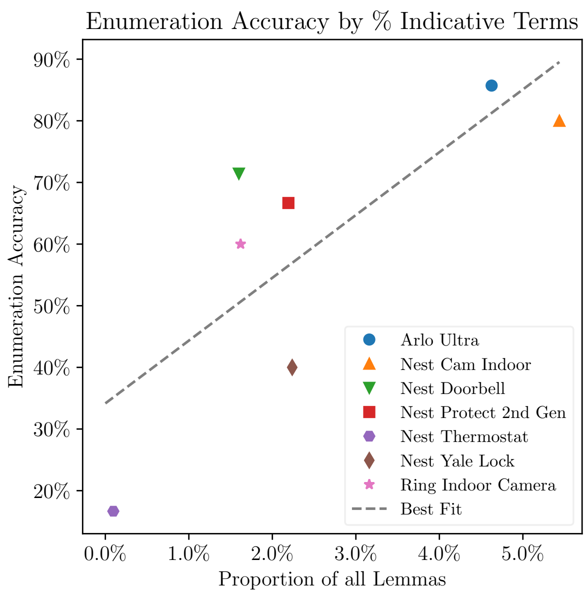


Figure 6.6: The proportion of corpus terms that are indicative correlated with transducer enumeration accuracy of device cognizant KTSM.

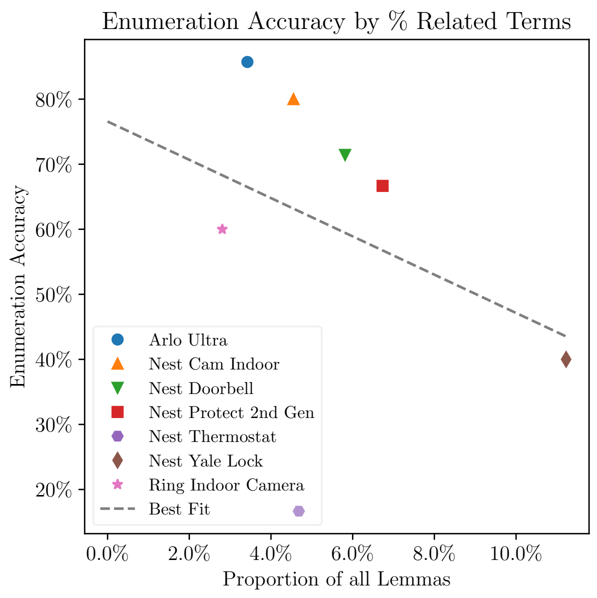


Figure 6.7: The proportion of corpus terms that are related correlated with transducer enumeration accuracy of device cognizant KTSM.

An interesting property that follows from this comparison is the correlation between a corpus' term class proportion and the enumeration accuracy for each device. Seen in Figure 6.6, as the proportion of indicative terms grows larger, the overall enumeration accuracy of the dcKTSM algorithm generally does as well. This is fairly intuitive, in that a set of vendor materials with a larger number of terms that more explicitly reference a particular transducer will better inform of the association of that transducer with the device. This means that a more refined ontology will contribute to an increased number of indicative terms and improve the enumeration accuracy.

Similarly, Figure 6.7 displays the negative relationship between proportion of lemmas that are related terms and the transducer enumeration accuracy. This relationship follows from that of indicative term proportion and transducer enumeration accuracy, as a higher proportion of related terms in a corpus is likely accompanied by a lower proportion of indicative terms in the corpus.

6.4 Proportion of Incorrect Transducer Matches

To evaluate the tendencies of our algorithms to incorrectly identify transducers, we also measure the proportion of transducers enumerated by an extraction algorithm that are not actually

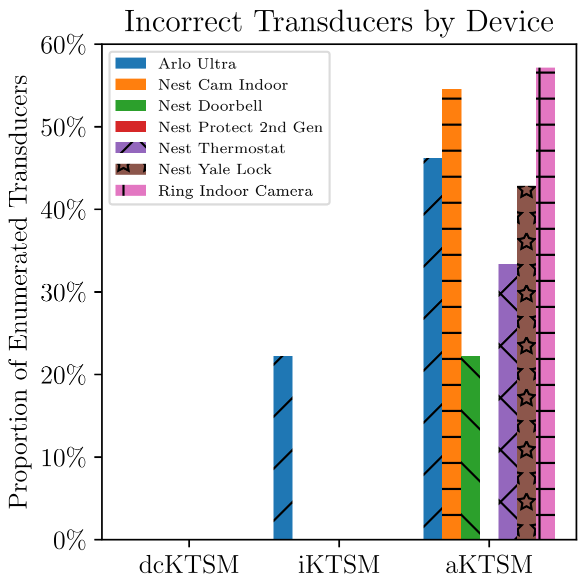


Figure 6.8: For each device, grouped by KTSM algorithm, the proportion of matching transducers that are incorrectly identified.

within the ground truth set. In other words, we measure the false discovery rates at which our algorithms enumerate transducers that are contained in the ontology, but not actually associated with the device.

Figure 6.8 shows the proportion of incorrect matches per device for each algorithm. The all-KTSM approach suffers from the highest proportions of incorrect matches across all devices, with an average of 36% of all transducer matches for each device. Indicative KTSM, on the other hand, averages only 3% incorrect transducers, and device cognizant KTSM does not incorrectly attribute any transducers to any of the evaluated devices.

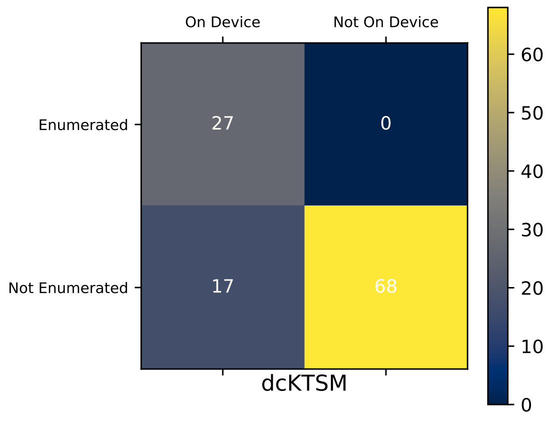
The advantage that comes from applying device cognizant KTSM on texts extracted from vendor materials is the fact that only the indicative terms that are associated with certain devices as defined in the ontology are applied for matching. This ensures that only the most relevant terms with the least ambiguity will be used to draw conclusions about the device's transducers. If device cognizant KTSM behaves in a way that is too restrictive and fails to enumerate transducers that indicative or all-KTSM can enumerate, it may be the case that the term sets used for matching are not thorough enough, or a sign that the ontology's representation of the device in question should be improved.

6.5 Precision, Recall, and F_1 Score

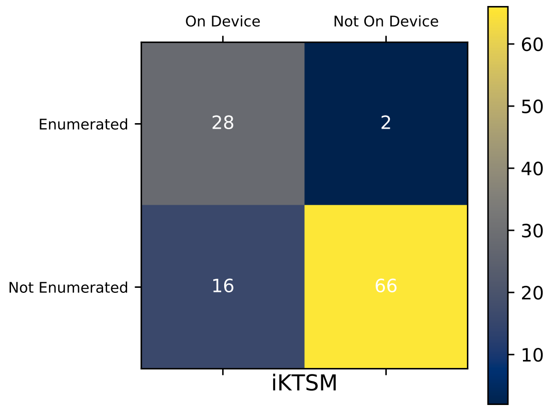
To better illustrate the trade-off between the enumeration accuracy and rate of incorrectly matched transducers for each algorithm, we present the precision, recall, and F_1 scores for our approach, modeling the extraction as a binary classification problem in which an algorithm correctly predicts the presence of a transducer that is part of the ground truth set of each device. For each algorithm, these composite scores are computed across all seven devices, meaning that the discrete number of true positives, true negatives, false positives, and false negatives are computed per-device before being summed across all devices.

Figure 6.9 displays a confusion matrix for each algorithm, which captures the enumerations of all transducers for each of the 7 evaluated devices. Note that the ontology that is used for the evaluation contains 16 total transducers, and that the number of true negatives is first computed separately for each evaluated device before being summed. These confusion matrices therefore represent the total numbers of correctly enumerated transducers, incorrectly enumerated transducers, transducers missed, and transducers that were correctly not enumerated for each algorithm.

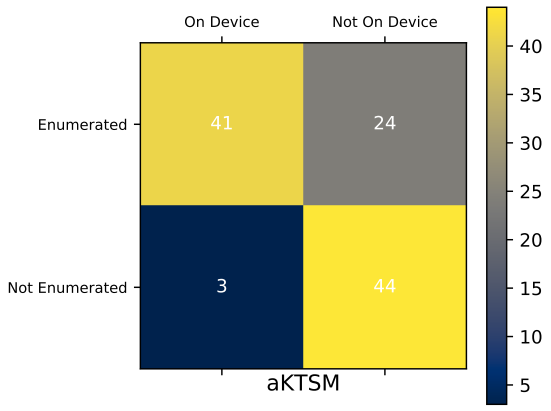
We compute the precision, recall, and F_1 score of each algorithm using the confusion matrices in Figure 6.9. The F_1 scores for the different algorithms are 0.761 for device cognizant KTSM, 0.757 for indicative KTSM, and 0.752 for all-KTSM. The more constrained algorithms, device cognizant and indicative KTSM, exhibit high precision with 1.00 and 0.933, respectively. However, the less-constrained alternative of all-KTSM suffers from a much lower precision of 0.631. At the same time, aKTSM has a much higher recall of 0.932 compared to the recall values of dcKTSM and iKTSM, which are 0.614 and 0.636, respectively. This indicates that the more constrained algorithms more consistently enumerate transducers correctly, but more often fail to enumerate all transducers.



(a) Device cognizant KTSM confusion matrix



(b) Indicative KTSM confusion matrix



(c) All-KTSM confusion matrix

Figure 6.9: Discrete confusion matrices for each algorithm, computed for all ontology transducers across the 7 evaluated devices.

Chapter 7

Discussion

7.1 Reliance on Vendor-Provided Materials

This work currently relies on the vendor-provided materials that are publicly available to derive the device capabilities. Missing information in those materials may result in an incomplete list of capabilities. To overcome this limitation, we intend to extend our approach to extract capabilities from the software specifications (e.g., device configurations, firmware) in addition to hardware specifications in the future.

7.2 Effects of Enumeration Accuracy

The obtained results on the enumeration accuracy of our solution in Section 6.3 indicate the accuracy of our matching algorithms in automatically enumerating device transducers from vendor materials. Some results clearly indicate that refinements to the ontology, and specifically the term sets of different transducers, should be made. Our solution allows for such refinements to take place iteratively, where an ontology can be augmented continuously with new vocabularies and knowledge of transducer-device relationships.

In one respect, the similar F_1 scores computed in Section 6.5 obfuscate the misclassification tendencies of the different algorithms and therefore the true nature of the trade-off relationship that exists between the algorithms. When considering the precision and recall of each algorithm, this relationship is more clear, and suggests that the choice of which algorithm to use is dependent on the goals of the user, organization, or application, as is the broadness of the transducer term sets. If the goal is to ensure that only transducers that are truly present be enumerated, then a more focused set of indicative terms should be used with device cognizant or indicative key term set matching. On the other hand, if the goal is to ensure that as many transducers as possible are enumerated, then all key term set matching with a more extensive dictionary of indicative and related terms should be used.

A point of interest going forward is to improve the transducer enumeration accuracy in an automated fashion, using a learning-based feedback loop. This could involve the discovery of new terms or alterations to the data model's representation of abstract device types or transducers.

7.3 Adapting to Other IoT Domains

Even though this thesis elaborates on the context of a smart home, our solution can potentially be adapted to other IoT domains (e.g., smart grid, smart health, autonomous vehicle). The main adaptation efforts would be to learn the ontology of the vendor materials of the application-specific devices and develop the extraction technique for those materials. The subsequent steps of our approach would remain the same.

Chapter 8

Applying Extracted Capabilities to Access Control

As our method produces a set of capabilities that are present on a device, a security or privacy application could utilize these facts about devices as attributes of a policy-driven access control system. In such a scheme, a user could define security policies that involve several attributes, including device location, time of day, and device capability, provided by our solution. We briefly explore in this section a model for such a system that is able to capture fine-grained security goals as specified by users within a smart home or administrators within an industrial or business setting.

The proposed model is based off of NIST Next Generation Access Control (NGAC) [28] and focuses on providing users with the ability to define policies that center around device capabilities being allowed to be in certain areas of an environment. In NGAC, policy components fall into hierarchical categories and are related to each other through relations, including associations for granting privileges and prohibitions for denying privileges. In our model, we consider policy components in terms of attributes of device capabilities which device instances can be associated with, and location-based environmental attributes (i.e., where a device may be located with the environment).

We consider a unary association for granting a single privilege, denoted $C_i - \{A\} - E_i$, which indicates that capabilities that fall under the capability attribute group C_i are allowed within environment locations that fall under the environmental group E_i. Similarly, we can explicitly specify that capabilities within a capability group C_i are not to be allowed within an environmental group E_i using a prohibition, which is denoted $C_i - X - E_i$.

Different environments or security goals require different default associations between capabilities and environmental attributes. We propose two general possibilities:

- **Exclusive "black-list" policies:** Policies which, by default, allow all capabilities within all environmental attributes. Users must explicitly specify exclusions to this default policy with prohibitions.

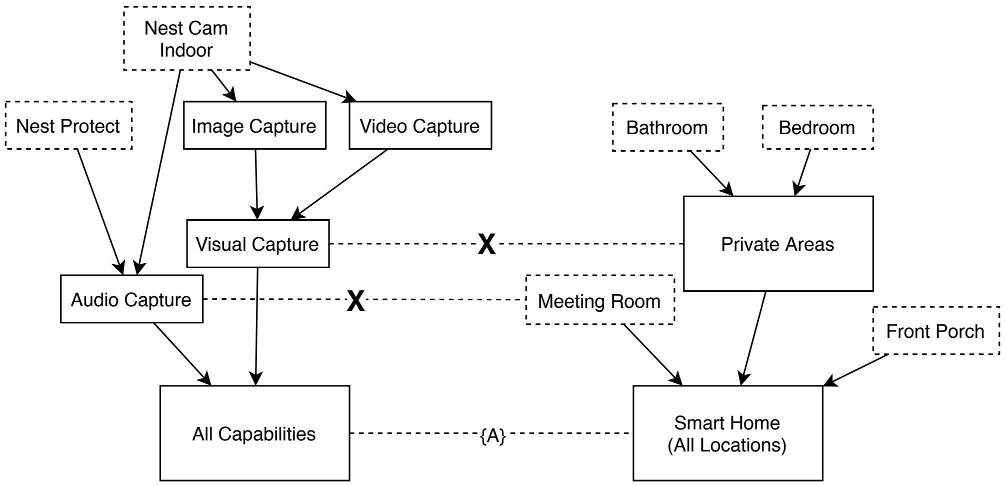


Figure 8.1: An example of an exclusive "black-list" access control policy, in which all device capability groups are considered to be allowed in all environmental groups, except when they are explicitly prohibited.

- **Inclusive "white-list" policies:** Policies which, by default, prohibit all capabilities within all environmental attributes. Users therefore must explicitly specify which capabilities should be allowed in which environmental attribute groups via allow associations.

Consider the example of a policy that specifies that no device that is capable of image or video capture should be allowed within private areas of a smart home, such as the bathroom and bedroom, and that no device that is capable of audio capture should be allowed within a specific room that is used for confidential meetings. Such a policy is modeled with our scheme and illustrated in Figure 8.1.

This policy could potentially be violated by a smart mirror that provides an image sensor being placed in the bathroom, or a smart smoke alarm such as the Nest Protect that includes a microphone being placed in the meeting room. This particular policy example can be considered as a "black-list" policy, where the default is to allow all capability groups within all environmental groups, and prohibitions are used to deny specific capability groups from specific environmental groups (including the entire smart home). The resulting policy is quite open, but not very complex and therefore may be well suited for typical smart home users who need only define a few critical exclusionary policies to ensure that their security goals are satisfied.

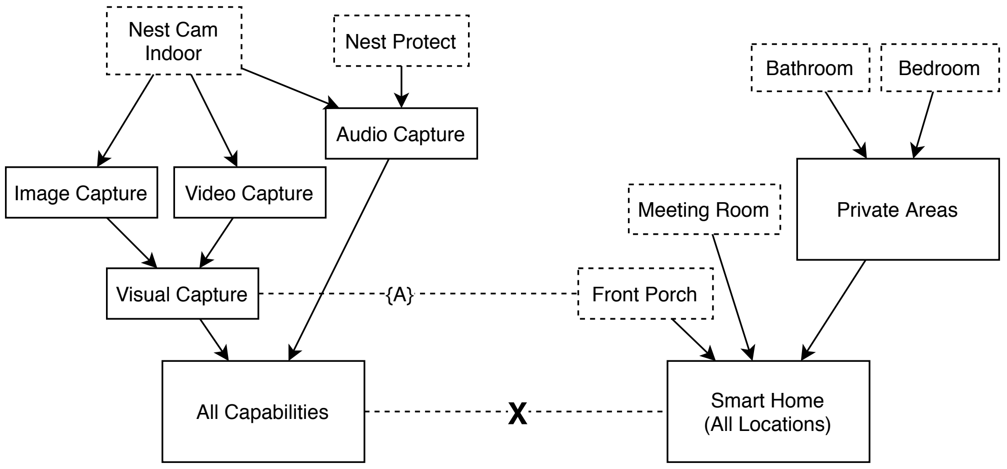


Figure 8.2: An example of an inclusive "white-list" access control policy, in which no device capability groups are considered to be allowed in all environmental groups, except when they are explicitly allowed with associations.

On the other hand, users or administrators with more stringent security goals may prefer the adoption of an inclusive "white-list" policy, which assumes that no capability group should be allowed in any environmental group, unless explicitly included by way of an appropriate association. Such a policy is illustrated in Figure 8.2, which specifies that devices with visual capture capabilities should be allowed on the front porch and the meeting room, and that devices with audio capture capabilities should be allowed in the meeting room as well. In this scheme, if a Nest Cam Indoor IoT camera is placed in a private area such as the bedroom or bathroom, the policy would be violated.

8.1 General Model

Generally, our policy scheme can be modeled as is illustrated in Figure 8.3. In this scheme, any capability group can be white or black-listed through a relation to any environmental attribute group. Depending on the policy, a device instance may be given its own relation to a specific environmental attribute group in order to ensure that the specific device is allowed or denied privileges within locations that fall under that environmental attribute group.

Capabilities enumerated with our extraction solution could be verified against user-defined security policies in this proposed model and therefore proactively affirm that users' IoT security goals

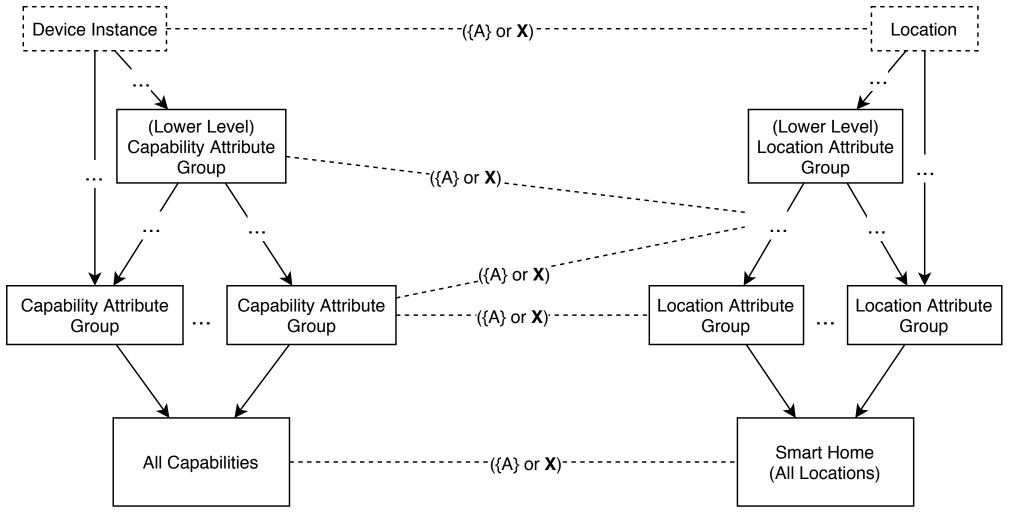

Device Instance
({A} or X)
Location
(Lower Level) Capability Attribute Group
(Lower Level) Location Attribute Group
({A} or X)
({A} or X)
Capability Attribute Group
Capability Attribute Group
({A} or X)
Location Attribute Group
Location Attribute Group
All Capabilities
({A} or X)
Smart Home (All Locations)

Figure 8.3: A general representation of the proposed access control model.

are satisfied. As discussed previously, many device vendors abstract away the actual functionalities of their devices with the language and format of their related vendor materials. Our capability model and extraction implementation can be applied to this proposed access control scheme to ensure that the security goals of users can be represented by policies that can be proactively verified without any prerequisite knowledge from users.

8.2 Policy Conflict Resolution

Note that, as is the case with NGAC policies, policy conflicts can come about. For example, the policy illustrated in Figure 8.1 specifies (from bottom to top), first that all capabilities are allowed in all smart home locations. However, one step lower in the hierarchy (higher in the figure) specifies that visual capture capabilities should not be permitted in private areas, which conflicts with the more general relation.

For a smart home domain, a method of conflict resolution that is most appropriate is prioritizing the more specific policy (in the previous example, the prohibition on visual capture capabilities). This way, users can specify more broad security goals that can be refined with finer-grained relations at the lower levels. There are alternative conflict resolution methods that could be better suited to different security goals, such as prioritizing relations that specify a particular environmental group. For example, ensuring that any relations that involve the “private areas” environmental group are prioritized over any others that conflict.

8.3 Extension to other Attributes

Our proposed access control model focuses on location-based attributes of the environment. However, this policy architecture could be well-suited for many other attributes of capabilities and ecosystems, such as time of operation, roles of users invoking the particular capability, or history of access or device behaviors. To achieve some of these more complex additions to the access control model, higher level concepts of NGAC, such as obligations, could be applied.

Chapter 9

Related Work

Research on IoT security has gained significant interest. These studies [5–12, 29–34] can be considered in a few different categories, including device fingerprinting, application monitoring, intrusion detection, and access control.

The existing device fingerprinting techniques [5–11] monitor and analyze the network traffic of IoT devices. More specifically, [5, 6] automatically discover and profile device behaviors by building machine learning models trained on network traffic according to their service (e.g., DNS, HTTP) and the semantic behaviors of devices (e.g., detected motion), respectively. Similar analysis is performed in Zhang *et al.* [7], where the fingerprints of a particular smart home devices are built using their network traffic. Other works [8–11] use similar techniques to automatically determine device identity or typical aggregate behaviors (as opposed to specific behaviors). Bezawada *et al.* [8] utilize machine learning to build behavior profiles based on network traffic for devices using the device category and device type. IoTSentinel [9], AuDI [10] and DeviceMien [11] use unsupervised learning to build models for individual device types based on network traffic captured during a device connection.

There exist several other security solutions (e.g., [31,33,35,36]) for smart homes. The existing application monitoring techniques [35, 36] run on source code of IoT applications and analyze these applications. More specifically, ContextIoT [35] and SmartAuth [36] offer permission-based systems to monitor an individual app. ProvThings [12] builds provenance graphs using security-critical APIs for IoT forensics. Soteria [37] and IoTGuard [33] verify security and safety policies by performing static and dynamic code analysis, respectively. Zhang et al. [31] monitor isolation-related properties among IoT devices through a virtual channel. Yang et al. [30] protect IoT devices from remote attacks by obfuscating them behind onion gateways.

The existing works suffer from several limitations as follows.

- Most of the solutions above rely on a great amount of inference, especially when considering encrypted network traffic. Many solutions either perform entropy analysis of encrypted traffic [8] or use only the features of network traffic that are not encrypted, such as TCP headers and other packet and flow metadata [5–7, 9–11]. Because of this inference, false positives and false negatives are a legitimate concern of these solutions.

- As most of the existing works rely on the application of inferential models (machine learning or otherwise), they are vulnerable to deceptive attacks, where an adversary may craft an attack that conforms to the model's expectation of legitimate traffic or behavior, thereby circumventing the model. On the other hand, an adversary could produce an attack that simply performs a denial-of-service attack by, for example, inundating the system with purposefully malicious traffic that does not conform to the model in order to overwhelm the model and prevent the processing of any legitimate traffic.

- These solutions cannot detect/prevent the critical safety or privacy implications that are not observable from the network traffic or application analyses, as they require access to either network traffic when devices are in operation, or application sources. These observations are also independent of other environmental attributes such as device location and therefore may not fully encapsulate security goals of consumers.

- Additionally, these related works do not present a proactive solution. Works that build profiles of devices can only do so when devices are present in an IoT environment and their traffic can be observed. Security solutions that analyze IoT applications for smart homes can provide insights on security flaws that applications have before they are in use, but they also require access to applications that may not be available.

This thesis, on the other hand, targets a different threat model where we extract an unambiguous representation of IoT device capabilities from their vendor material that will facilitate evaluating potential security threats even before a device is installed. These capabilities are generic and are therefore independent of device implementation and vendor. Further, our capability model represents capabilities in a way that is more in line with the innate security goals and policies of consumers. Our extraction methodology also does not require visibility of network traffic or estimate device behavior heuristically.

Chapter 10

Conclusion

10.1 Summary

With the growing popularity of IoT, the necessity of ensuring its security continues becoming more important than ever. To that end, existing works rely on observable data (e.g., network traffic, sensor data, etc.) and therefore suffer from several security and potential privacy concerns. We proposed an approach to enumerate IoT device capabilities from their related vendor-provided materials in order to ensure that consumers or security applications acting on their behalf can be proactively informed, in a clear and unambiguous way, of what a device is able to do.

More specifically, we introduced an IoT device capability model that can represent device capabilities in an unambiguous way and used a manual review process to build an ontology that encoded learned information about vendor-provided documentation materials. We implemented and evaluated a novel set of algorithms that can be used to extract transducer information from vendor materials and map that information to device capabilities. Additionally, we presented a scheme for fine-grained and expressive user-defined attribute-based access control that leverages the capability information that our extraction approach provides.

10.2 Limitations and Future Research Directions

Our current work relies only on vendor provided materials that are publicly available and may be missing some information. In the future, we plan to augment our approach with information from software specifications, device configurations, and additional materials extracted from device firmware. Additionally, improvements to automate the building of the vendor material ontology are an area of particular interest for future work, where new terms or other contextual items provided by materials could be discovered automatically. Our future work also includes adapting our methodology to other IoT ecosystems including the smart grid, smart health, autonomous vehicles, and industrial IoT. We also leave the verification of our proposed access control scheme, as well as additional extensions to it described in Section 8.3 as future work.

Bibliography

[1] Statista market forecast. Internet of Things (IoT) connected devices installed base worldwide from 2015 to 2025, 2016.

[2] Nest Protect 2nd Gen - Installation and Tech Specs - Google Store. https://store.google.com/us/product/nest_protect_2nd_gen_specs.

[3] 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. of USENIX*, pages 255–272, Baltimore, MD, August 2018.

[4] Nest Cam Indoor - Installation and Tech Specs - Google Store. https://store.google.com/us/product/nest_cam_specs.

[5] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, and J. Lloret. Network Traffic Classifier With Convolutional and Recurrent Neural Networks for Internet of Things. *IEEE Access*, 5:18042–18050, 2017.

[6] 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 *Proc. of WiSec*, pages 128–138, 2019.

[7] Wei Zhang, Yan Meng, Yugeng Liu, Xiaokuan Zhang, Yinqian Zhang, and Haojin Zhu. HoMonit: Monitoring Smart Home Apps from Encrypted Traffic. In *Proc. of CCS*, pages 1074–1088, 2018.

[8] Bruhadeshwar Bezawada, Maalvika Bachani, Jordan Peterson, Hossein Shirazi, Indrakshi Ray, and Indrajit Ray. Behavioral Fingerprinting of IoT Devices. In *Proc. of ASHES*, pages 41–50, 2018.

[9] M. Miettinen, S. Marchal, I. Hafeez, N. Asokan, A. Sadeghi, and S. Tarkoma. IoT SENTINEL: Automated Device-Type Identification for Security Enforcement in IoT. In *Proc. of ICDCS*, pages 2177–2184, June 2017.

[10] S. Marchal, M. Miettinen, T. D. Nguyen, A. Sadeghi, and N. Asokan. AuDI: Toward Autonomous IoT Device-Type Identification Using Periodic Communication. *IEEE Journal on Selected Areas in Communications*, 37(6):1402–1412, June 2019.

[11] Jorge Ortiz, Catherine Crawford, and Franck Le. DeviceMien: Network Device Behavior Modeling for Identifying Unknown IoT Devices. In *Proc. of IoTDI*, pages 106–117, Montreal, Quebec, Canada, 2019.

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

[13] Simon Birnbach, Simon Eberz, and Ivan Martinovic. Peeves: Physical Event Verification in Smart Homes. In *Proceedings of CCS*, pages 1455–1467. ACM, 2019.

[14] Statista market forecast. Smart home – United States. https://www.statista.com/outlook/279/109/smart-home/united-states, 2019.

[15] Nest Cam Indoor - Home Security Camera - Google Store. https://store.google.com/us/product/nest_cam.

[16] Technical specifications for Nest cameras and video doorbells - Google Nest Help. https://support.google.com/googlenest/answer/9259110.

[17] Camera API | Nest Developers. https://developers.nest.com/reference/api-camera.

[18] Arlo Ultra | HD Security Camera | Wireless Camera System. https://www.arlo.com/en-us/products/arlo-ultra/default.aspx.

[19] V.K. Shen, D.W. Siderius, W.P. Krekelberg, and H.W. Hatch. Considerations for Managing Internet of Things (IoT) Cybersecurity and Privacy Risks. Technical report, National Institute of Standards and Technology, June 2019.

[20] Nest x Yale Lock - Key-Free Smart Deadbolt - Google Store. https://store.google.com/us/product/nest_x_yale_lock.

[21] Nest Hello Video Doorbell - Know Who's Knocking - Google Store. https://store.google.com/us/product/nest_hello_doorbell.

[22] Nest Learning Thermostat - Installation and Tech Specs - Google Store. https://store.google.com/us/product/nest_learning_thermostat_3rd_gen_specs.

[23] Indoor Cam | Indoor Security Cameras | Ring. https://shop.ring.com/products/mini-indoor-security-camera.

[24] Requests. Requests: HTTP for humans. https://requests.readthedocs.io/en/master/.

[25] beautifulsoup. beautifulsoup4 4.8.2. https://pypi.org/project/beautifulsoup4/.

[26] spaCy. spaCy: Industrial-strength natural language processing in python. https://spacy.io/.

[27] V. Singh. Replace or Retrieve Keywords In Documents at Scale. *ArXiv e-prints*, October 2017. Provided by the SAO/NASA Astrophysics Data System.

[28] David Ferraiolo, Ramaswamy Chandramouli, Rick Kuhn, and Vincent Hu. Extensible Access Control Markup Language (XACML) and Next Generation Access Control (NGAC). In *Proceedings of ABAC*, page 13–24, 2016.

[29] Jiwon Choi, Hayoung Jeoung, Jihun Kim, Youngjoo Ko, Wonup Jung, Hanjun Kim, and Jong Kim. Detecting and identifying faulty IoT devices in smart home with context extraction. In *IEEE DSN*, 2018.

[30] Lei Yang, Chris Seasholtz, Bo Luo, and Fengjun Li. Hide your hackable smart home from remote attacks: The multipath onion IoT Gateways. In *ESORICS*, 2018.

[31] Yang Zhang and Jun-liang Chen. Modeling virtual channel to enforce runtime properties for IoT services. In *ICC*, 2017.

[32] Smriti Bhatt, Farhan Patwa, and Ravi Sandhu. An access control framework for cloud-enabled wearable Internet of Things. In *CIC*, 2017.

[33] Z. Berkay Celik, Gang Tan, and Patrick D McDaniel. IoTGuard: Dynamic Enforcement of Security and Safety Policy in Commodity IoT. In *NDSS*, 2019.

[34] Martin Serror, Martin Henze, Sacha Hack, Marko Schuba, and Klaus Wehrle. Towards in-network security for smart homes. In *ARES*, 2018.

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

[36] 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 *USENIX Security*, 2017.

[37] Z. Berkay Celik, Patrick McDaniel, and Gang Tan. Soteria: Automated IoT Safety and Security Analysis. In *USENIX ATC*, 2018.