Proactively Extracting IoT Device Capabilities: An Application to Smart Homes

Andy Dolan[1], Indrakshi Ray[1], and Suryadipta Majumdar[2]

[1] Computer Science, Colorado State University, USA
[2] Information Security and Digital Forensics, University at Albany, USA

Abstract. 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 the 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. We apply our approach to the context of a smart home and evaluate its accuracy and efficiency on the devices from three different vendors.

1 Introduction

The popularity of IoT devices is gaining momentum (e.g., projections of 75.44 billion devices worldwide by 2025 [18]). This large ecosystem is comprised of a variety of devices that are being used in diverse environments including healthcare, industrial control, and homes. Manufacturers emphasize certain features and characteristics of the IoT devices and often abstract away their actual design complexity and functionalities from the user. Many IoT devices are equipped with an extended set of sensors and actuators which allows them to perform different functionalities. For example, a smart light with a microphone and motion detector can possibly perform far more than just light sensing.

Such abstraction and extended (and in many cases hidden) functionalities of an IoT device result in a blind spot for the consumers and leave an IoT system vulnerable to various security and privacy threats. Installing the above-mentioned smart light necessitates that the consumer understand its potential security and privacy consequences. This requires the consumer to study its design specifications to find out what sensors it possess. Furthermore, she must have the insights to realize the security and privacy consequences of having a microphone and motion detector in a light, and determine if any of those consequences violate the 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.

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 we know the capabilities of a device, we can check if any of 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 breaches and can take adaptive measures to mitigate that risk.

Several works [21,24,35,12,23,22,25,33,14] 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 [35,12] 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 [14]) 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 design specifications. We first define the notion of device capability in the context of IoT. Second, we extract the transducer (e.g., sensors and actuators) information for each device using vendor-provided specification materials. Third, we identify the capabilities of a device by deriving the capabilities of each sensor and actuator of that device. 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 [19]) and evaluate its efficiency and accuracy. The main contributions of this paper 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 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.

The remainder of the paper is organized as follows. Section 2 summarizes related work. Section 3 provides background on vendor materials. Section 4 presents our methodology. Section 5 describes its implementation. Section 6 presents the evaluation results. Section 7 concludes the paper.

2 Related Work

Research on IoT security has gained significant interest. These studies [21,24,35,12,23,22,25,33,17,34,36,13,16,28,27]) are categorized into device fingerprinting, application monitoring, intrusion detection, and access control.

The existing device fingerprinting techniques [21,24,35,12,23,22,25] monitor and analyze network traffic in IoT. More specifically, [21,24] 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.* [35], where the fingerprints of a particular smart home device are built using its network traffic. Other works (e.g., [12,23,22,25]) use similar techniques to automatically determine device identity or typical aggregate behaviors (as opposed to specific behavior). Bezawada *et al.* [12] utilize machine learning to build behavior profiles based on network traffic for devices using the device category and device type. IoTSentinel [23], AuDI [22] and DeviceMien [25] 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., [20,32,16,36]) for smart homes. The existing application monitoring techniques (e.g., [20,32]) run on source code of IoT applications and analyze these applications. More specifically, ContextIoT [20] and SmartAuth [32] offer permission-based systems to monitor an individual app. ProvThings [33] builds provenance graphs using security-critical APIs for IoT forensics. Soteria [15] and IoTGuard [16] verify security and safety policies by performing static and dynamic code analysis, respectively. Zhang et al. [36] monitor isolation-related properties among IoT devices through a virtual channel. Yang et al. [34] protect IoT devices from remote attacks by hiding them inside onion gateways.

Limitations of Existing Work. First, 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 [12] or use only the unencrypted features of network traffic such as TCP headers and other packet and flow metadata [21,24,35,22,23,25]. Because of this inference, false positives and false negatives are a legitimate concern of these solutions. Second, 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. An attack at the other end of this spectrum would be to simply conduct a denial-of-service attack by, for example, inundating the system with purposefully malicious traffic to overwhelm the model and prevent the processing of any legitimate traffic. Third, these related works cannot detect/prevent the critical safety or privacy implications that are not observable from the network traffic.

Our paper, on the other hand, is complementary to those existing works, and targets a different threat model where we extract IoT device capabilities from their design specifications that will facilitate evaluating potential security threats even before a device is installed.

3 Vendor Materials

3.1 Vendor Material Description

We consider vendor materials including product webpages, technical specifications, and developer documentations which are publicly available and contain aspects of the specifications (sensors, actuators, or related features) of a device.

Product Webpages. Product webpages are official marketing pages from which a consumer can purchase the product, and contain the summary information about a device. For instance, Google has an online store for its smart home products (e.g., [4]). These pages can be an initial source of information about a smart home device and its specifications.

Technical Specifications. Technical specification pages provide details about a device and its hardware specifications. For instance, Google has a technical specification page for its smart home products (e.g., [5]). This work considers these technical specification pages as one of the most significant sources of information about a device's hardware components.

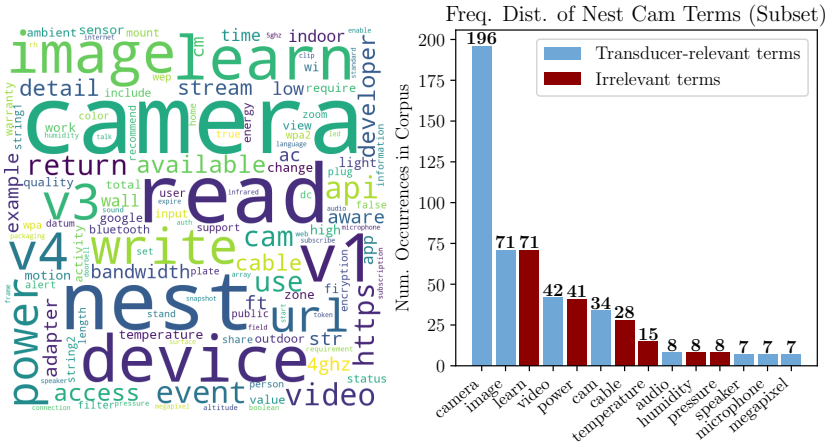
Developer Documentations. Developer documentations provide information to developers who create applications for the smart home devices. Even though these materials are intended for application developers, they can be used as a source for the extraction of information about the hardware and capabilities of a device.

3.2 Investigation on the Real-World Vendor Materials

Analysis of the Vendor Materials. We analyze the contents of several vendor materials by leveraging natural language processing techniques. These analyses result in insights on the challenges that come with the extraction of vendor materials, which is illustrated through the following examples.

Figure 1a 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. 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 [4,5,10,2]. 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 1b illustrates the frequency distribution of only a subset of notable terms.



(a) (b)

Fig. 1. (a) The term frequencies for the corpus of vendor materials on the Nest Cam Indoor, visualized as a word cloud. (b) 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.

Additionally, term frequency-inverse document frequency (TF-IDF) calculations are also performed within individual corpora for a device. Specifically, each separate vendor material for a device is treated as an article in the corpus, and the TF-IDF metric is computed for an individual corpus. Overall, TF-IDF fails to find terms that are most indicative of a device's transducers and capabilities due to their infrequent appearances. We also apply TF-IDF on each section of a vendor material as an individual document. However, this technique also does not produce any conclusive results.

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 Standaridized 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. This implies significant effort to learn those different templates to enable their extraction.
- *Brevity of the Materials.* Vendor materials are usually 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 the 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 Jargons.* Each vendor tends to use their own set of terminologies for their devices. Mainly due to their business policy, vendors craft languages around what information they believe is the most useful to or well-received by the consumer, and include terminology that may be unique to only their line of products. Accordingly, learning the vocabularies used for various vendors and then normalizing them to infer their capabilities presents additional challenges.
- *Interpreting Visual Contents.* Several contexts (e.g. device type) of a material is visually represented and is therefore very difficult to encode 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. Therefore, text processing alone becomes insufficient in those cases.
- *Distributed Materials.* The information about a device is distributed over various materials (e.g., product webpage, technical specifications, user manual, and developer documentation), and it is essential to obtain information from as many different materials as possible, normalize their formats, and extract capabilities for the most thorough extraction.

4 Methodology

We first present our threat model and the assumptions of our approach, followed by the overview, and finally the details.

4.1 Threat Model

We focus on smart homes, an IoT application, 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 from 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 paper. In this work, we derive the device capabilities from the vendor-provided materials that are publicly available. In this paper, the impact of negation in the language of

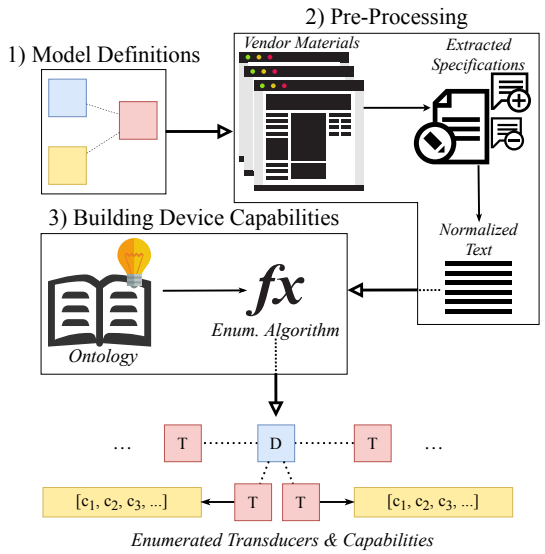


Fig. 2. An overview of our methodology, including 1) development of the model, 2) pre-processing of vendor materials, and 3) applying an ontology to extracted specifications by way of an automatic enumeration function.

this materials is not considered in extracting capabilities. Therefore, any missing information about a device in those materials may affect the effectiveness of our approach.

4.2 Overview

Figure 2 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 the example of a motion-activated smart camera throughout.

[**Step 1: Defining Capability Model for IoT Devices**] We first define IoT devices, then define their transducers (e.g., sensors and actuators), and finally define the capability models that map transducers to their set of capabilities. (See Section 4.3).

[**Step 2: Normalizing Specifications from Vendor Materials**] 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 4.4).

[**Step 3: Building IoT Device Capabilities**] We first build the ontology of the device specifications, 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 4.5).

4.3 Defining Capability Models for IoT Devices

This work defines a *transducer* as a sensor or actuator (partly inspired by the definitions in NIST 8228 [29]). A *sensor* holds the core functionality of sensing or measuring various aspects of a physical environment and converting it 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 various physical actions (e.g., emitting light, producing sound, actuating a lock to toggle its state).

The set of all transducers T is partitioned into two sets, where S is the set of all sensors and A is the set of all actuators, where $T = S \cup A$ and $S \cap A = \emptyset$. A capability of a sensor is denoted as c_{si} and the 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, where $m \neq n$ and $s_m, s_n \in S$, their set of capabilities may overlap.

Definition 1. [Transducer]: *A* transducer t_i *is either a sensor* s_i *or an actuator* a_i*. That is, if* $t_i = s_i$*, then* $t_i \neq a_i$*. Also, if* $t_i = a_i$*, then* $t_i \neq s_i$*. Each sensor* s_i *and each actuator* a_i *consists of a non-zero finite set of capabilities, denoted as* $s_i = \{c_{s1}, c_{s2}, \ldots, c_{sp}\}$ *and* $a_i = \{c_{a1}, c_{a2}, \ldots, c_{aq}\}$*. Also* $s_i \neq \{\}$ *and* $a_i \neq \{\}$*.*

Definition 2. [Device:] *A* device *is 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 number of transducers in* D_i *equals* $n + m$*.*

Multiple devices may have common sensors or actuators. For example, smart cameras and smart video doorbells both have a camera sensor. Two devices D_r and D_s shown below have common sensor s_3 and common actuator a_2.
$D_r = S_1 \cup A_1 = \{s_1, s_2, s_3, a_1, a_2\}$; $D_s = S_2 \cup A_2 = \{s_3, s_4, a_2, a_3\}$.

Definition 3. [Device Capability]: *A* capability of device D_i *is a function the device can perform. The set of capabilities for* D_i *is computed as the union of the set of capabilities of the transducers comprising the device.*

Multiple devices can have common capabilities. For example, a smart camera has an image sensor and therefore holds the capability of capturing an image. On the other hand, a smart light has both light and motion sensors; therefore it holds the capabilities of sensing lights and detecting motion. In the following, the two devices D_p and D_q have common capabilities c_{s3} and c_{a2}:
$D_p = \{c_{s1}, c_{s2}, c_{s3}, c_{a2}, c_{a3}\}$; $D_q = \{c_{s3}, c_{s4}, c_{s5}, c_{a1}, c_{a2}\}$.

4.4 Normalizing Specifications from Vendor Materials

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.

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. This work extracts the vendor materials in HTML and/or text formats.

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. We parameterize this step so that specific sections of a page can be extracted. For example, the technical specifications page for a smart camera may contain page elements unrelated to the device, such as navigation links, or even additional specification information for similar products (e.g., video doorbells). With our parameterized method, we are able to extract only the specification information for the smart camera. We also tailor the parameters to specific vendor pages; these parameters are often reusable, as web design under a single vendor is often homogeneous.

Normalizing the Data. To ensure that the contents extracted from the vendor materials are best suited for the transducer enumeration technique, our approach normalizes those contents by removing elements that are not critical to the enumeration process, including punctuation, non-alphanumeric, and non-white-space characters, as well as "stop words" (i.e., common articles and prepositions in English). The term case and plurality are also normalized through lemmatization, a process of linguistics that simply involves homogenizing different inflections of the same term to the dictionary form of the term. 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, a motion activated camera's marketing page may contain the following text: *"with the camera's array of motion sensors, video will be recorded automatically"*. After this step, the same string will read *"camera array motion sensor video record automatically"*.

4.5 Building IoT Device Capabilities

We now describe how to derive the device capabilities after pre-processing.

Building the Ontology. Our approach understands the language and structure of the vendor materials, and then builds an ontology. The ontology will contain an understanding of the terminologies used to refer to specific components of a device. It is possible to include other information as well that can be derived from the vendor products.

Enumerating Transducers. To enumerate device transducers using the above-mentioned ontology, we devise three algorithms, namely, ranked key term set matching (rKTSM), unranked key term set matching (KTSM), and unguided key term set matching, where *key terms* are one or more words related to a

transducer. These algorithms use two types of key terms: indicative terms and related terms. The *indicative terms* are unambiguously indicative of the presence of a transducer. The *related terms* are related to a transducer, but may be more ambiguous, and hence are not sufficient in drawing conclusions about its presence. In the case of a motion-activated camera, the term "camera" is considered indicative of an image sensor, while the term "video" is considered related to an image sensor. We describe each algorithm below, in which the notation "x[y]" indicates the mapping or membership of a field y in x.

– *Ranked Key Term Set Matching (rKTSM) Algorithm:* Algorithm 1 first filters the key term sets for different abstract 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 "get_matches" function extracts any matching key terms in the provided set that are contained within the input corpus. 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.

Algorithm 1: Ranked Key Term Set Matching (rKTSM)

```
1  best_score ← 0
2  best_indicative ← 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)
```

The rKTSM 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 the entire corpus, the

presence of related terms such as “audio” or “voice” can help bolster confidence when concluding that a microphone transducer is present.

– *Unranked Key Term Set Matching (KTSM) Algorithm:* Algorithm 2 uses only the indicative terms without ranking term sets. 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: Unranked Key Term Set Matching (KTSM)

```
all_indicative_terms ← ⋃_d d[transducers][indicative_terms]
ind_matches ← get_matches(all_indicative_terms, corpus)
transducer_identifiers ← reverse_map(ind_matches)
```

– *Unguided Key Term Set Matching:* Additionally, we consider a completely unguided KTSM alrogithm, shown in Algorithm 3, that performs the same matching as unranked KTSM, but also matches on the related terms. We consider this algorithm to be the least focused and exact, as it attempts to match using an entire vocabulary.

Algorithm 3: Unguided Key Term Set Matching

```
all_indicative_terms ← ⋃_d d[transducers][indicative_terms]
all_related_terms ← ⋃_d d[transducers][related_terms]
all_terms ← all_indicative_terms + all_related_terms
all_matches ← get_matches(all_terms, corpus)
transducer_identifiers ← reverse_map(all_matches)
```

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 fit into the model that is described in Section 4.3.

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 Section 4.3. The capabilities contained in the final output set represent a device’s functionality unambiguously. Table 1 provides examples of the outputs of our methodology.

5 Implementation

We build the ontology of vendor materials for seven smart home products: Arlo Ultra Camera, Nest Cam Indoor, Nest Hello Doorbell, Nest Protect, Nest Learning Thermostat, Nest X Yale Lock, and Ring Indoor Camera [1,4,6,8,7,9,3].

Table 1. An excerpt of outputs from our approach.

Device	Category	Transducer	Atomic Capabilities
Arlo Ultra [1]	**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 [5]	**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 [8]	**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 [9]	**Sensors**	Light sensor	Detect light
		Touch Sensor	Detect (capacitive) contact
	Actuators	Lock	Lock and unlock door
		Speaker	Produce sound
		LED Light	Produce light

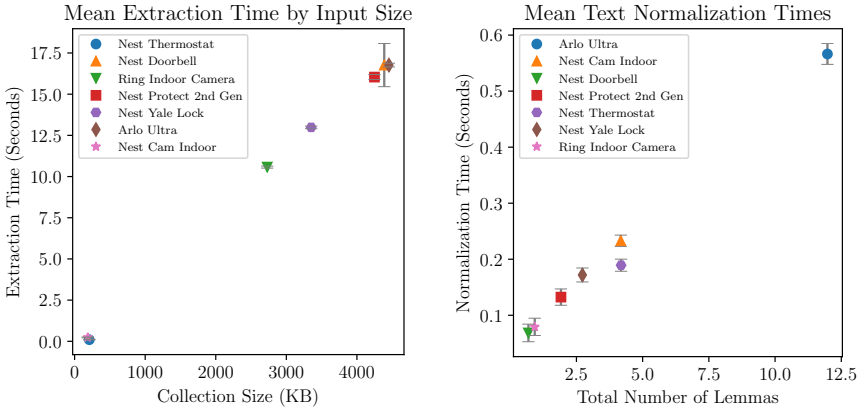
Our pre-processing step is implemented (in Python) to fetch the product web pages directly over the network via the requests library [26] by way of their URI, or to read pages fetched previously and saved locally. To extract the textual content from those pages, we utilize the BeautifulSoup package [11] 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 elements in web pages (i.e., HTML), which is the current format for most vendor materials. However, if some vendor materials only display the 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 [31].

The KTSM algorithms described in Section 4.5 are also implemented as Python functions that take a corpus as input from which to enumerate transducers. To encode the ontology created during the manual review process (as described in Appendix A), a data model is created 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. 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 [30] to perform key term matching. In the case of ranked KTSM, the number of matches is used to determine the best abstract device type. In unranked and unguided KTSM, only the matching step takes place with all indicative terms, and with all indicative and related terms, respectively. Any matches are mapped to their associated transducer's identifier automatically (enabled by the Python implementation of flashtext).

6 Evaluation

This section discusses the performance of our implemented solution, which is used to extract enumerations of transducers for the seven devices. 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 its efficiency, the enumeration accuracy for each device, and the proportion of incorrect transducer matches for each device.



(a) (b)

Fig. 3. (a) Average text extraction time of the vendor materials for different devices by their size on disk, computed over 5 trials. (b) Average text normalization time by the total number of lemmas in a corpus, computed over 5 trials.

Efficiency. The goal of the first set of our 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 3a, 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 3a indicates that extraction time scales with the size linearly.

For most evaluated devices, the times required to perform the text normalization step, displayed in Figure 3b, are negligible when compared to those of the extraction step. Consistently, across all devices, the text normalization step is performed in less than a second, increasing slightly with the total number of lemmas extracted.

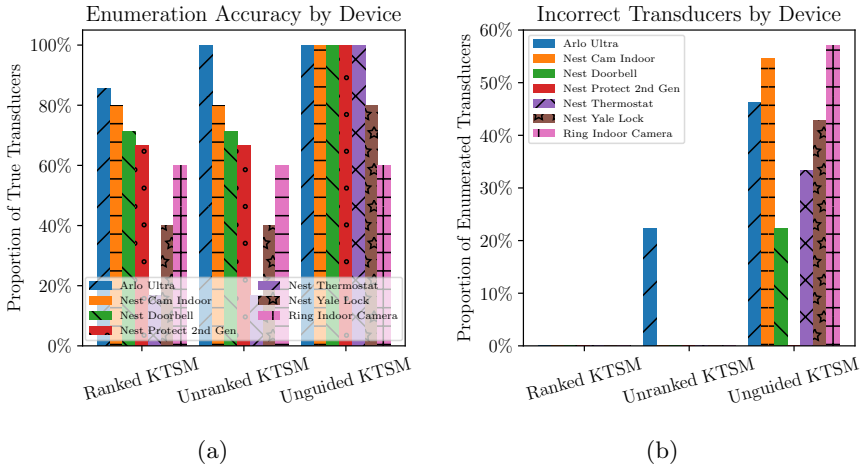


Fig. 4. For each device, grouped by KTSM algorithm (a) the proportion of ground truth transducers that are correctly enumerated, and (b) the proportion of matching transducers that are incorrectly identified.

We do not analyze the impact of the automatic enumeration of device transducers from normalized texts, due to the efficiency of the tree-based flashtext algorithm for keyword extraction which we use. Additionally, we do not consider the mapping from enumerated transducers to their capabilities as having any impact on efficiency, 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.

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 computed as the ratio between the number of these ground truth transducers that were correctly identified by an extraction approach to the total number of ground truth transducers. Figure 4a displays the results of our enumerations on the seven different devices from three different vendors, grouped by the enumeration algorithms.

Figure 4a shows that both the ranked and unranked KTSM algorithms provide similar proportions of correctly identified transducers, averaging about 60%

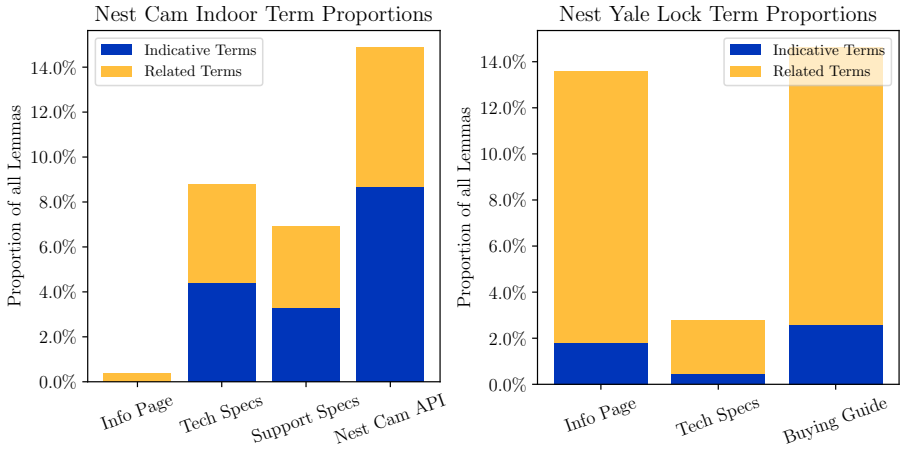


Fig. 5. A comparison of the proportions of indicative and related terms per document for (a) the Nest Cam Indoor [5] and (b) the Nest X Yale Lock [9].

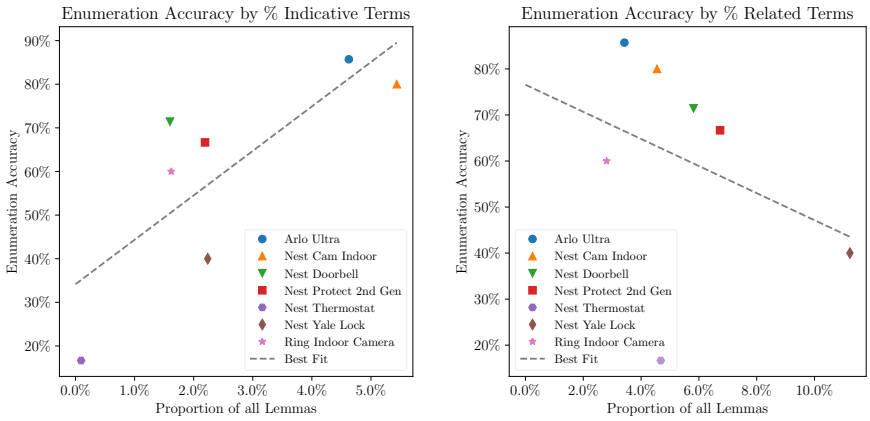
and 62%, respectively, of ground truth transducers identified among all devices with a standard deviation of 24% and 27%, respectively.

For each extraction approach, there is a large disparity between the transducer enumeration rate for the Nest Cam Indoor and the Nest Yale Lock. We present a comparison of these two devices in Figure 5, showing the proportion of indicative and related terms for each document per device. An interesting property that follows from this comparison is the correlation between this property and the enumeration accuracy for each device. As can be seen in Figure 6a, as the proportion of indicative terms grows larger, the overall enumeration accuracy of the rKTSM 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 a reader 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 6b displays the negative relationship between proportion of lemmas that are related terms and the transducer enumeration rate. This relationship follows from that of indicative terms and transducer enumeration rate, as a higher proportion of related terms in a corpus is likely accompanied by a lower proportion of indicative terms in the corpus.

Proportion of Incorrect Transducer Matches. To evaluate the tendencies of our algorithms to incorrectly identify transducers, we also measure the proportion of matched transducers that are not within the ground truth set. In other words, we measure the rates at which our algorithms enumerate transducers that are not actually associated with the device.

Figure 4b shows the proportion of incorrect matches. The unguided 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. Un-



(a) By proportion of indicative terms (b) By proportion of related terms

Fig. 6. The proportion of corpus terms that are (a) indicative and (b) related correlated with transducer enumeration rate of ranked KTSM.

ranked KTSM, on the other hand, averages only 3% incorrect transducers, and ranked KTSM does not incorrectly attribute any transducers to any of the new evaluated devices.

The advantage that comes from applying ranked 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 ranked KTSM behaves in a way that is too restrictive and fails to enumerate transducers that unranked or unguided 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.

7 Conclusion and Future Work

With the growing popularity of IoT, the necessity of ensuring its security becomes important than ever. We proposed a proactive approach to extract IoT device capabilities from their design specifications to verify their potential threats even before a device is installed. More specifically, we defined the notion of device capability in the context of IoT, extracted the transducer information for each device using vendor-provided design specifications and finally identified the capabilities of a device.

Our current work relies only on vendor provided materials that are publicly available and may be missing some information. In future, we plan to augment our approach with information from software specifications, device configurations, and firmware versions. Our future work also includes adapting our methodology to other IoT applications including smart grid, smart health, and autonomous vehicle.

Acknowledgement

The authors thank the anonymous reviewers for their comments. 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.

References

1. Arlo ultra — hd security camera — wireless camera system, https://www.arlo.com/en-us/products/arlo-ultra/default.aspx
2. Camera API — nest developers, https://developers.nest.com/reference/api-camera
3. Indoor cam — indoor security cameras — ring, https://shop.ring.com/products/mini-indoor-security-camera
4. Nest cam indoor - home security camera - google store, https://store.google.com/us/product/nest_cam
5. Nest cam indoor - installation and tech specs - google store, https://store.google.com/us/product/nest_cam_specs
6. Nest hello video doorbell - know who's knocking - google store, https://store.google.com/us/product/nest_hello_doorbell
7. Nest learning thermostat - installation and tech specs - google store, https://store.google.com/us/product/nest_learning_thermostat_3rd_gen_specs
8. Nest protect 2ng gen - installation and tech specs - google store, https://store.google.com/us/product/nest_protect_2nd_gen_specs
9. Nest x yale lock - key-free smart deadbolt - google store, https://store.google.com/us/product/nest_x_yale_lock
10. Technical specifications for nest cameras and video doorbells - google nest help, https://support.google.com/googlenest/answer/9259110
11. beautifulsoup: beautifulsoup4 4.8.2, available at: https://pypi.org/project/beautifulsoup4/
12. Bezawada, B., Bachani, M., Peterson, J., Shirazi, H., Ray, I., Ray, I.: Behavioral Fingerprinting of IoT Devices. In: Proc. of ASHES. pp. 41–50 (2018)
13. Bhatt, S., Patwa, F., Sandhu, R.: An access control framework for cloud-enabled wearable Internet of Things. In: CIC (2017)
14. Birnbach, S., Eberz, S., Martinovic, I.: Peeves: Physical Event Verification in Smart Homes. In: Proceedings of CCS. pp. 1455–1467. ACM (2019)
15. Celik, Z.B., McDaniel, P., Tan, G.: Soteria: Automated IoT Safety and Security Analysis. In: USENIX ATC (2018)
16. Celik, Z.B., Tan, G., McDaniel, P.D.: IoTGuard: Dynamic Enforcement of Security and Safety Policy in Commodity IoT. In: NDSS (2019)
17. Choi, J., Jeoung, H., Kim, J., Ko, Y., Jung, W., Kim, H., Kim, J.: Detecting and identifying faulty IoT devices in smart home with context extraction. In: IEEE DSN (2018)
18. market forecast, S.: Internet of Things (IoT) connected devices installed base worldwide from 2015 to 2025 (2016), https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/

19. market forecast, S.: Smart home – United States (2019), https://www.statista.com/outlook/279/109/smart-home/united-states
20. Jia, Y.J., Chen, Q.A., Wang, S., Rahmati, A., Fernandes, E., Mao, Z.M., Prakash, A., Unviersity, S.J.: ContexIoT: Towards Providing Contextual Integrity to Appified IoT Platforms. In: NDSS (2017)
21. Lopez-Martin, M., Carro, B., Sanchez-Esguevillas, A., Lloret, J.: Network Traffic Classifier With Convolutional and Recurrent Neural Networks for Internet of Things. IEEE Access **5**, 18042–18050 (2017)
22. Marchal, S., Miettinen, M., Nguyen, T.D., Sadeghi, A., Asokan, N.: AuDI: Toward Autonomous IoT Device-Type Identification Using Periodic Communication. IEEE Journal on Selected Areas in Communications **37**(6), 1402–1412 (June 2019)
23. Miettinen, M., Marchal, S., Hafeez, I., Asokan, N., Sadeghi, A., Tarkoma, S.: IoT SENTINEL: Automated Device-Type Identification for Security Enforcement in IoT. In: Proc. of ICDCS. pp. 2177–2184 (June 2017)
24. OConnor, T., Mohamed, R., Miettinen, M., Enck, W., Reaves, B., Sadeghi, A.R.: HomeSnitch: Behavior Transparency and Control for Smart Home IoT Devices. In: Proc. of WiSec. pp. 128–138 (2019)
25. Ortiz, J., Crawford, C., Le, F.: DeviceMien: Network Device Behavior Modeling for Identifying Unknown IoT Devices. In: Proc. of IoTDI. pp. 106–117. Montreal, Quebec, Canada (2019)
26. Requests: Requests: HTTP for humans, available at: https://requests.readthedocs.io/en/master/
27. Román-Castro, R., López, J., Gritzalis, S.: Evolution and trends in IoT security. Computer **51**(7), 16–25 (2018)
28. Serror, M., Henze, M., Hack, S., Schuba, M., Wehrle, K.: Towards in-network security for smart homes. In: ARES (2018)
29. Shen, V., Siderius, D., Krekelberg, W., Hatch, H.: Considerations for Managing Internet of Things (IoT) Cybersecurity and Privacy Risks. Tech. rep., National Institute of Standards and Technology (June 2019), https://doi.org/10.6028/NIST.IR.8228
30. Singh, V.: Replace or Retrieve Keywords In Documents at Scale. ArXiv e-prints (Oct 2017), http://adsabs.harvard.edu/abs/2017arXiv171100046S, provided by the SAO/NASA Astrophysics Data System
31. spaCy: spaCy: Industrial-strength natural language processing in python, available at: https://spacy.io/
32. Tian, Y., Zhang, N., Lin, Y.H., Wang, X., Ur, B., Guo, X., Tague, P.: SmartAuth: User-Centered Authorization for the Internet of Things. In: USENIX Security (2017)
33. Wang, Q., Hassan, W.U., Bates, A., Gunter, C.: Fear and Logging in the Internet of Things. In: NDSS (2018)
34. Yang, L., Seasholtz, C., Luo, B., Li, F.: Hide your hackable smart home from remote attacks: The multipath onion IoT Gateways. In: ESORICS (2018)
35. Zhang, W., Meng, Y., Liu, Y., Zhang, X., Zhang, Y., Zhu, H.: HoMonit: Monitoring Smart Home Apps from Encrypted Traffic. In: Proc. of CCS. pp. 1074–1088 (2018)
36. Zhang, Y., Chen, J.l.: Modeling virtual channel to enforce runtime properties for IoT services. In: ICC (2017)

A Study on Vendor materials

Reviewing vendor materials manually is important for understanding any inferences required in drawing conclusions about a device's transducers and capabil-

ities. 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) on the device. During this process, any insights, inferences, or context clues that are used to make this interpretation are also captured. The most important of these features that are key terms that act as indicators of the presence of a particular transducer or capability. In some cases, key terms are only indicative of a transducer or capability when considered in conjunction with another key term.

Table 2. Obtained ontology for the Nest Cam Indoor

Category	Transducer	Capabilities
Sensors	Image Sensor	Capture Image, Capture Video, Detect Light
	Microphone	Capture Sound
Actuators	Speaker	Produce Sound
	LED Light	Produce Light
	Infrared Light	Produce Infrared Light

For example, during the manual review of the Nest Cam Indoor's technical specifications on the Google Nest support forums [10], simple terms such as "camera" are indicative of the presence of an image sensor. On the other hand, a term like "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. A sample outcome for the Google Nest Cam Indoor is summarized in Table 2.

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

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

Fig. 7. Different sections of the Nest Cam Indoor technical specifications page [10], 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.

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 7.

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 reviewer to conceive of these possibilities is dependent on their technical background knowledge.