Computer Networks 192 (2021) 108040

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Survey paper

A survey on IoT platforms: Communication, security, and privacy perspectives

Leonardo Babun [a,*], Kyle Denney [a], Z. Berkay Celik [b], Patrick McDaniel [c], A. Selcuk Uluagac [a]

[a] Cyber-Physical Systems Security Lab (CSL), College of Electrical and Computer Engineering, Florida International University, Miami, FL, USA
[b] Department of Computer Science, Purdue University, West Lafayette, IN, USA
[c] Systems and Internet Infrastructure Security (SIIS) Lab, Department of Computer Science and Engineering, Penn State University, University Park, PA, USA

ARTICLE INFO

Keywords:
Privacy
IoT platforms
Security
Communications
Internet of Things

ABSTRACT

The Internet of Things (IoT) redefines the way how commodity and industrial tasks are performed every day. The integration of sensors, lightweight computation, and the proliferation of different wireless technologies on IoT platforms enable human beings to easily interact with their surrounding physical world thoroughly. With the recent rise of IoT, several different *IoT platforms* have been introduced for researchers and developers to ease the management and control of various IoT devices. In general, the IoT platforms act as a bridge between core IoT functionalities and users by providing APIs. Due to their wide variety of applications, IoT platforms are mostly unique in their architectures and designs. Thus, IoT administrators, developers, and researchers (i.e., IoT users) are challenged with substantial configuration differences in the proper configuration, implementation, and protection of the IoT solutions. In this survey, we conduct an in-depth analysis of popular IoT platforms from different application domains. More specifically, we define a comprehensive evaluation framework that considers seven different technical comparison criteria: (1) topology design, (2) programming languages, (3) third-party support, (4) extended protocol support, (5) event handling, (6) security, and (7) privacy. Then, we use the framework to evaluate the different IoT platforms highlighting their distinguishing attributes on communications, security, and privacy. First, we describe the communication protocols supported by the different IoT platforms surveyed. Then, rather than uncovering novel threats affecting IoT, we aim to analyze how the different IoT platforms handle security and privacy vulnerabilities affecting the most common security services of confidentiality, integrity, availability, and access control. Further, we present possible solutions that these platforms could implement to strengthen security and privacy within the IoT solution. Finally, we discuss the advantages and disadvantages of every IoT platform, so IoT administrators, developers, and researchers (i.e., IoT users) can make an informed decision on the use of specific platforms to implement their IoT solutions. To the best of our knowledge, this is the first comprehensive survey to evaluate different IoT platforms using the criteria defined in this work.

1. Introduction

The Internet of Things (IoT) is evolving the way we live and help perform commodity and industrial tasks. For this, IoT integrates new functionalities to commonplace objects. For instance, smart locks can communicate with smart fire alarms to unlock the doors in the event of a fire. The integration of sensors and computation power into everyday objects allows for an easy interaction with the physical world; something that never has been achieved at this scale with traditional systems before. For example, it is now possible for objects like a clock alarm and a coffeemaker to work together and prepare our morning coffee right before we wake up. In general, IoT covers a wide variety of fields: from autonomous vehicles, smart homes, to even modern agriculture — everything around us is being integrated into the IoT realm at an incredibly rapid pace. Specifically, by 2025, the IoT is projected to number over 75 billion connected devices [1].

Meanwhile, different *IoT platforms* provide the programmatic tools necessary to integrate a rich set of functionalities via numerous APIs. Due to the high diversity of devices, applications, and interests in IoT, dozens of these IoT platforms are active today. IoT application developers or administrators have the challenging task of choosing an IoT platform that best fits their needs based on, for instance, the cost, number, and type of available APIs, the programming language used, devices supported, etc. IoT developers, administrators, and researchers (i.e., IoT users) are traditionally concerned with ensuring that their

* Corresponding author.
E-mail address: lbabu002@fiu.edu (L. Babun).

https://doi.org/10.1016/j.comnet.2021.108040
Received 1 April 2020; Received in revised form 11 January 2021; Accepted 21 March 2021
Available online 29 March 2021

applications and systems work as intended. However, concerns related to security and privacy in IoT are rapidly rising in the community and among users [2–5]. In addition to making sure the IoT solutions perform as needed, an IoT user should either be a security/privacy expert or trust the tools provided by the IoT platform in order to have a secure IoT environment, neither of which is necessarily guaranteed easily. In this context, several security solutions for IoT have been proposed in the past [6–11]. However, all these solutions typically focus on specific IoT architecture and fail to provide comprehensive solutions that protect devices and apps considering different platforms simultaneously.

In this survey, we address two primary concerns of an IoT user[1]: (1) what evaluation criteria may be used while selecting the most suitable IoT platform for a specific need and (2) what communication, security, and privacy mechanisms are supported by the analyzed IoT platforms. We start by detailing popular IoT platforms from different application domains, highlighting their key strengths and weaknesses. Through this, we establish a generalized method for evaluating future IoT platforms using specific criteria. This allows IoT users to find the best fit for their IoT needs accurately. From there, we dive into IoT communications, security, and privacy. We first detail different communication protocols supported by IoT platforms, which may impact the specific application and security of IoT solutions. Then, we discuss current security and privacy mechanisms supported by IoT platforms. Specifically, we discuss issues in confidentiality, integrity, availability, and privacy, which still exist in modern IoT platforms. We aim to cover the concerns that IoT users should be aware of, but also highlight where the community needs further research.

The main goal of this survey is to provide IoT users with adequate information in the state-of-the-art of IoT platforms to smartly choose the ideal platform to adopt for their IoT solutions. We also provide the research and the academic community with areas where further research and development are needed, specifically in IoT security and privacy. First, we describe the communication protocols supported by the different IoT platforms surveyed here. Then, rather than uncovering novel threats affecting IoT, we aim to analyze how the different IoT platforms handle security and privacy vulnerabilities affecting the most common security services of confidentiality, integrity, availability, and access control. Further, we present possible solutions that these platforms could implement to strengthen security and privacy within the IoT solution. We define a comprehensive evaluation framework that considers seven different technical comparison criteria: (1) topology design, (2) programming languages, (3) third-party support, (4) extended protocol support, (5) event handling, (6) security, and (7) privacy. With our evaluation framework, we help IoT users to make informed decisions regarding their design choices to more effectively implement their IoT solutions. Further, we contribute to IoT researchers and developers to more efficiently conduct their research and development efforts by (1) overcoming current challenges imposed by the high diversity of IoT [12] and (2) by identifying missing or weak functionalities that may impact the communication capabilities and the security and privacy in current IoT solutions.

Contributions: The contributions of this manuscript are as follows:

- We study and analyze the most popular, state-of-the-art IoT platforms to identify what design choices best fit an IoT user's needs.
- We introduce an evaluation framework with general criteria to study, compare, and evaluate different IoT platforms. The evaluation framework may also be used as a baseline to examine future, different-purpose IoT platforms.
- We provide comparative tables that easily highlight our findings. IoT users may use these tables as a guideline to select which IoT platform is more suitable for their specific IoT applications.
- We also identify key areas of research and development still needed in the realm of IoT communications, security, and privacy.

Organization: The rest of the paper is organized as follows. Section 2 presents the background information relevant to IoT platforms. Also, it defines terminology used throughout this manuscript. We introduce the evaluated IoT platforms in Section 3. Section 4 defines the evaluation criteria we use to compare the different IoT platforms. Then, in Section 5, we conduct the evaluation of each IoT platform based on the proposed criteria. We also compare how the criteria vary throughout each platform as platforms adopt different techniques to meet user and application demands. We summarize our findings and discussions in Section 6. Finally, Section 7 presents the related work and Section 8 concludes the manuscript, and discusses future work.

2. Background

In this section, we overview a general architecture of an IoT solution regarding the platform used. From there, we define the key features that we use throughout the manuscript.

2.1. The Internet of Things (IoT)

The proliferation of wireless technology and the increasing efficiency of small, embedded-systems [13,14] led to a technological convergence that is nowadays known as the Internet of Things (IoT). An IoT device is a generic term for any small computing device that can communicate with other devices. Atzori et al. note in 2010 that, "the main strength of the IoT idea is the high impact it will have on several aspects of everyday life and behavior of potential users" [15]. As of this writing, IoT devices have moved into our homes [6,8,16], allowed cars to drive themselves [12], and have generally automated life around us. Countless numbers of IoT devices work "hand-to-hand" to automate these processes in ways never thought possible.

2.2. IoT Solutions

Fig. 1 depicts the general architecture of an IoT solution. In this manuscript, we use the term *IoT solution* to describe a fully working setup for IoT. Due to the particularities in IoT, describing IoT just as a network of devices would not provide a complete vision as just networking IoT devices together would not necessarily make them operate in the way and for the purpose they were designed. An IoT solution demands synchronization among many IoT devices (and, in many cases, cloud servers) that require specialized applications, communication protocols, and data processing capabilities that must also be considered. Therefore, we use the term IoT solution to bundle all of this specialization under one colloquial term. Specifically, an IoT solution that we describe here includes four interdependent layers: (1) IoT Devices, (2) Communication Protocols, (3) Data Processing, and (4) IoT Applications (Fig. 1). From bottom to top, the lowest layer of the discussed IoT solution includes the IoT devices. These devices, depending on their application purpose, combine specific sensors and actuators that permit perceiving the surrounding physical environment and acting based on pre-programmed event handlers. For instance, the motion sensor detects the presence of the user and triggers the smart light to turn on. Going up a layer, a collection of communication protocols permit the devices to communicate and create network links among each other and, in several cases, with cloud-based servers. Originally, traditional communication protocols like WiFi and Bluetooth were directly integrated into the IoT realm. However, considering that IoT's tasks highly differ from applications of traditional computing systems, new communication protocols such as ZigBee and Z-Wave have been utilized for IoT. The goal of these protocols is to guarantee the high connectivity required among IoT devices and systems while reducing the power consumption and increasing reliability. Finally, at

[1] We utilize the term IoT user to reference IoT developers, administrators, and researchers.

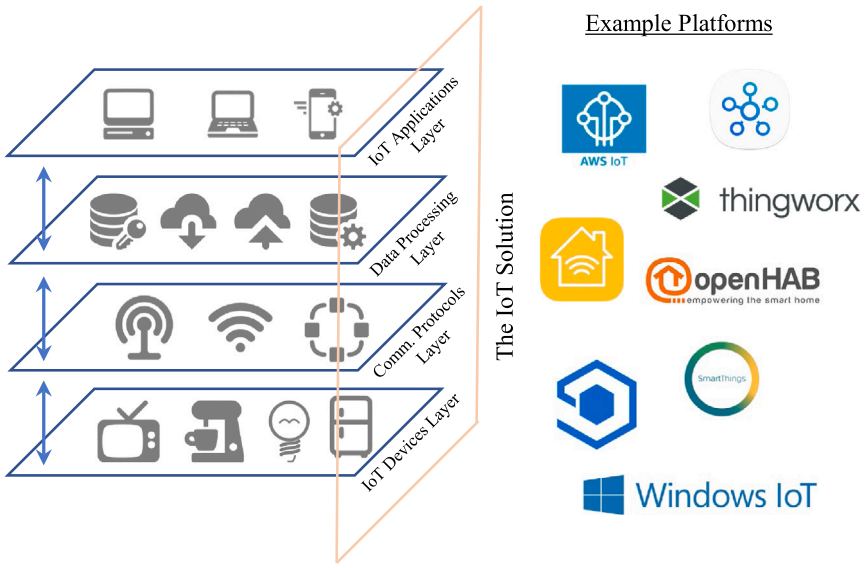


Fig. 1. Different IoT platforms integrate four layers of functionalities within an IoT solution.

the top of an IoT solution, Data Processing and IoT Application layers permit the analysis of the physical data collected by IoT devices and provide interfaces so users can manage the IoT solution and interpret the collected data. We provide additional details below:

IoT Devices. As mentioned before, IoT devices are equipped with embedded sensors, actuators, processors, and transceivers that permit the full interaction with the physical world. There are two main processes in which IoT devices play a main role. First, sensors detect changes to their physical surroundings and collect these changes as IoT data. Generally, this data provides valuable information regarding specific events of interest to the IoT solution. In fact, automation in IoT is achieved via sharing the IoT data with other devices, central processing units (i.e., a hub), or cloud-based servers to trigger specific events (e.g., a temperature sensor for reporting the air conditioner (AC) that the house needs cooling). Second, once a trigger event is detected, IoT devices are capable of influencing the physical environment using specific actuators (i.e., trigger-action interactions). For instance, from the previous example, after sensing the need for cooling the house, the smart thermostat changes the AC state from off to on.

Communication Protocols. Connectivity is a crucial element of an IoT solution. Thus, to guarantee the reliability of the IoT communications, many protocols can be used within the same IoT implementation to accommodate for constraints of the environment (e.g., Bluetooth, WiFi, Zigbee, Z-Wave, etc.). Some of these constraints are related to the physical setup (i.e., the distance between devices may impose the use of long-range communication protocols), the specific IoT task performed (i.e., real-time applications require higher connectivity capabilities), and the device's computing resources (i.e., power-restrained devices may impose the use of low-power communication protocols like Bluetooth Low Energy (BLE)). To overcome these communication challenges, standardization organizations, IEEE or the IETF, have worked on different IoT-specific communication protocols like the IEEE 802.15.4e, 6LoWPAN, and LoRa [12,17].

Data Processing. This layer provides the capacity for data analysis and control in an IoT solution. Through different programming paradigms and development, IoT solutions implement data processing capabilities necessary to manage the devices and their interactions while also enabling crucial functions such as data collection, control, and interoperability [9]. For instance, with data processing, an IoT solution can verify and execute the AC trigger-action interactions explained above. Also, data processing provides the means for IoT application development. To implement IoT-specific tasks, IoT developers integrate APIs and web interfaces into their apps, which allows for customization and fine-grained tuning of critical operations in the IoT solution. For instance, Samsung SmartThings offers the class *asynchttp_v1* class via an API to implement asynchronous HTTPS calls within an IoT solution [7,8].

IoT Applications. They allow the user to engage with an IoT solution, set control values, monitor the execution of critical tasks, and receive real-time information about the different IoT solution states. IoT apps implement features that enable the user to see real-time data of her IoT solution and react to the data. For instance, an application for a smart-home would allow the user to turn on her IoT devices, read current state data such as the temperature in the house, and schedule events such as turning on the lights when she arrives home.

Finally, within an IoT solution, the IoT platforms work as a cyber 'bridge' to enable functionality across these four different layers (Fig. 1). The IoT platforms that we evaluate in this manuscript provide the software architecture that permits all of these layers to interoperate and enable the specific tasks of the IoT solution. These platforms provide the necessary code, and implementation capabilities to (1) interact with supported IoT devices, (2) allow IoT devices to communicate with each other, (3) process data collected from end-devices, and finally (4) allow humans to interact with and understand this data. However, due to the high diversity in IoT, every platform provides specific tools to enable the IoT solution. This well-known heterogeneity challenges the integration of devices and software from different platforms into a single IoT solution. In addition, every platform uses different approaches to guarantee the connectivity among devices and servers, the implementation of security mechanisms to protect the IoT solution from cyber attackers, and the protection of the privacy-sensitive information from users and systems. In these scenarios, IoT users are challenged with the convoluted task of selecting the IoT platforms that best fit their needs. In the remaining sections of this survey, we highlight how the design choices of different popular IoT platforms affect the functionality of each layer and the IoT solution as a whole.

3. The IoT Platforms

In this section, we introduce some of the most popular and widespread IoT platforms among the industry and developer, research, and academic communities as of the writing of this survey. We present the criteria that we use to study, compare, and evaluate the IoT

platforms, highlighting specific features that consider the potential design needs of the IoT users. As noted earlier, for the purpose of this work, an IoT user refers to any developer, administrator, and researcher who actively utilize IoT resources to build, monitor, or perform analysis on IoT solutions.

3.1. Consumer IoT vs. Enterprise IoT

There exist several different criteria to effectively classify IoT platforms. The most common one split IoT devices and platforms into two main groups: Enterprise IoT and Consumer IoT. In this case, the classification criteria take into consideration the (1) functionality and context in which IoT devices are being utilized, (2) the desired interoperability, (3) granularity and scalability, (4) security and privacy requirements, and (5) the cost [18,19].

- *Functionality and Context*: In terms of functionality, IoT platforms from both groups (i.e., consumer and enterprise) may be able to assume similar applications. However, other variables such as cost and implementation complexity can vary significantly. These variables define the context in which an IoT platform is utilized. In general, consumer IoT (e.g., SmartThings, OpenHAB) is designed to provide users with peace-of-mind, plug-and-play implementation capabilities. The commodity IoT devices are designed so that the regular users should be able to undertake the entire implementation/configuration process by themselves. On the other hand, the devices from the enterprise IoT (e.g., Control4, Lutron, Creston) requires more professional installation, and the platforms or the vendors would spend more dedicated resources on training and licensing their specialized technicians [20–22].
- *Interoperability*: Consumer (or commodity) IoT is typically designed to facilitate the interconnection among devices from different manufactures and platforms. That way, the users may implement the IoT solutions using the devices of their choice (out of thousands of available options) or that would better-fit their preferences, needs, or budget. This interoperability of the commodity IoT devices allows for new functionalities and extended automation in some cases. For instance, Amazon Alexa can connect to devices from different manufactures and act as a controlling entity within a heterogeneous IoT solution, facilitating the configuration and monitoring of several devices for the users. Differently, the enterprise IoT solutions are more homogeneous in terms of device diversity and usually conform to a single manufacturer/vendor. While the consumer IoT generally utilizes open-source software and protocols to monitor and control their systems, the enterprise IoT is known to use proprietary protocols that require licensing.
- *Granularity and Scalability*: As mentioned before, the user of the consumer IoT has the option of selecting among thousands of different devices available in the market. However, every one of these devices typically provides specific functionality (e.g., light control, temperature control), and manufacturers tend to design specialized devices for specific applications. In these scenarios, the user must be, at least, capable of selecting the right devices for her specific application. However, one great advantage that the enterprise IoT platforms offer is that they are designed with the "comprehensive-solution" concept in mind. That is, the enterprise IoTs' manufacturers offer a wide range of devices and controllers that bring added capability to the solution and are capable of assuming any "smart" functionality. Finally, the enterprise solutions are meant to be used for scalable applications that require the support of, on the one side, complex services like audio/video transmission and, on the other side, several IoT "nodes"[2] simultaneously.
- *Security and Privacy Requirements*: Although protecting IoT systems and the privacy of the users constitute a concern for all the IoT platforms, there exist some differences in terms of requirements between the consumer and enterprise IoT. For the consumer IoT, most researchers and experts agree that the vulnerabilities affecting the privacy represent a higher threat to the regular user [7–9,16]. Due to the nature of the personalized commodity systems, attackers may take advantage of such vulnerabilities to fingerprint [23] user behavior or steal privacy-related information. However, the enterprise IoT is mainly utilized in professional and industrial environments where vulnerabilities may give hackers full access to critical infrastructure like the smart grid [13,14]. Thus, the enterprise IoT usually require more secure and higher standards compared to the consumer systems.
- *Price*: A direct consequence of the differences between consumer and enterprise IoT is also reflected in the budget. While the regular user may spend hundreds of dollars to acquire all the necessary devices to implement the IoT solution, the enterprise version would require tens of thousands of dollars to be spent on expensive equipment, proprietary software and protocols, licensing, and professional installation and support.

3.2. A functionality-aware selection of IoT Platforms

In the following, we introduce the different IoT platforms considered in this work. Because criteria to rank IoT platforms can be very diverse, we choose specific IoT platforms to evaluate based on two selection criteria that are key to the IoT user considered here: (1) *Platform Independence* and (2) *Platform Support*. Also, we aim to integrate IoT platforms that target different application domains (e.g., commodity IoT, industry, agriculture, etc.) into this survey. We include IoT platforms from different application domains to align better with the diversity of IoT. Also, by offering comparison criteria that can be applied to platforms from different domains, a broader spectrum of IoT users will benefit from this work. In Section 4, we detail the specific comparison criteria that allow us to analyze such a diverse set of IoT platforms. In the following, we detail the selection criteria that we used to analyze the IoT platforms. As a result of this analysis process, we select the top-eight platforms to be included in this survey.

- *Platform Independence*: The IoT platform must be able to implement a full IoT solution (regarding the specific domain) without any additional support from other platforms. That is, platforms that help automate IoT solutions are not included since they require another IoT platform to function. For example, C3 IoT platform [24] is not included as it runs on top of Amazon's IoT platform.
- *Platform Support*: The IoT platform must have the ability to thrive a well-designed, well-maintained support infrastructure, designed specifically for the IoT users considered here (i.e., developers, administrators, researchers). With over 450 IoT platform providers [25], it would be impossible for all of them to survive long-term. For this reason, we focus on evaluating platforms with a significant community (i.e., large open-source platforms) or company support (e.g., Microsoft, Apple, Samsung). Hence, IoT platforms with better support normally control the highest market share in terms of active devices and applications for their specific domain. For instance, as of this writing Samsung SmartThings platform dominates the market with the highest share of devices and applications [26,27].

In Table 1, we detail the selection criteria of the top-eight platforms considered here. We also include additional information such as their distribution models (e.g., payment models, open/closed sourced) as well as an estimation of the current number of IoT devices and applications supported by the specific IoT platform as of this writing. An interesting finding is that home-automation platforms such as

[2] In this specific case, the term IoT node refers for instance, to the different audio/video stations (e.g., display, controllers) within a solution.

Table 1
Selection criteria that led to the selection and inclusion of top-eight IoT platforms into this survey. We define as "extension" any handcrafted IoT app (different from the official app) that can be used to control IoT devices. Figures are as of February 2020.

IoT Platform overview				
Platform	Payment model	Open/Close sourced	Devices	Applications
openHAB [28]	Freeware	Open	1500+	310+ extensions
SmartThings [29]	Purchase devices, premium services	Open	390+	1 official; 500+ extensions
HomeKit [30]	Purchase devices, developer access	Closed	280+	1 official; 300+ extensions
Windows IoT [31]	Commercial agreements	Open	100+	1 official; 200+ extensions
FarmBeats [32]	Not open to public	Closed	100+	1 official
AWS IoT [33]	Subscription per device	Open	800+	28 official partners
ThingWorx [34]	Pay for IoT data	Open	1200+	4 official
Watson IoT [35]	Pay for IoT data	Closed	600+	1 official

SmartThings and HomeKit tend to have fewer compatible devices than general-purpose/industrial IoT platforms like ThingWorx and Watson IoT. Contrarily, the home automation platforms have many applications/extensions,[3] while the industrial platforms usually consider one main application or outsource functionality to official partners. Our observation for this outcome is that the need for plug-and-play and personalization at home lead to fewer devices, but more customization capabilities. On the other hand, the industry needs generalization and variety, which leads to a vast diversity of devices that can be used across many industrial domains (e.g., health, agriculture, factories), but one application that can be tuned to fit the customer's needs. In the following, we provide some overview analysis of the selected IoT platforms.

OpenHAB. OpenHAB is a Java-based smart home platform that implements an open-source solution to the Eclipse SmartHome framework [28]. While Eclipse SmartHome is the core functionality of a smart home environment, OpenHAB constitutes the full implementation that uses Apache Karaf and Eclipse Equinox for the Open Service Gateway Initiative (OSGi) runtime. In the OpenHab architecture, *things* are the physical entities connected to the solution. These things provide *channels* that represent passive entities that amount to what a thing offers. *items* represent the virtual back-end to things. They have a state and represent the functionality used by the application. Finally, *links* establish relationships between Things and Items, the physical connection between the two. Through these entities (i.e., things, items, links), the OpenHAB platform facilitates developers to define rules to build automated tasks. With this generic architecture, OpenHAB promotes the idea of vendor-neutrality and encourages devices from any vendor to be implemented in the OpenHAB platform. Rules in OpenHaB are written in Domain Specific Language (DSL), which is derived from Xbase programming language. Finally, OpenHAB rules use three different types of triggers to react to changes in the environment. *Event-based* triggers listen to commands from Items, *time-based* triggers respond to special times, and *system-based* triggers run with certain system states.

Samsung SmartThings. SmartThings is a proprietary home automation platform implemented by Samsung [29]. Following a producer/-consumer paradigm, SmartThings connects a large array of sensors and actuators to a central Hub and from there to the cloud-based server on the Internet. As a proprietary framework, much of the code generation and deployment are hidden via specific APIs to developers, which makes the development of SmartThings apps more user-friendly. SmartThings applications (i.e., SmartApps) are written in a dynamic, object-oriented Groovy language for the Hub or the cloud and then configured using mobile applications (i.e., controller device). The SmartThings design architecture includes two different abstractions [29]: (1) *capabilities*, which implement the event and actions for a device and (2) *intelligence*, that is used to develop an application for a specific type of device, regardless of the connection protocol or the manufacturer. Similar to OpenHAB, SmartApps are event-driven. The permission control allows the programmer to specify program inputs such as devices- and user-defined inputs. The developer then uses the event handlers to subscribe to a specific device or user-defined (e.g., location-related) events.

Apple HomeKit. HomeKit is also a proprietary home automation platform that performs full integration with Apple products [30]. Programming for this platform requires membership to the iOS Developer Program and can either be done in Swift or Objective C, two of Apple's preferred coding platforms. Built from iOS, HomeKit uses a centralized application to communicate with HomeKit accessories (i.e., IoT devices). HomeKit promotes the idea of simplicity to the user by hosting most of the programming in the back-end to automate installation for the user. Unlike previously discussed platforms, Apple HomeKit does not require a centralized gateway or Hub to guarantee communication between devices. In this case, users and different devices can communicate using Siri and HomeKit applications. From the architectural point of view, the main component of a HomeKit network is the *accessory* (i.e., device). These accessories are defined by *services*, which represent what an accessory can do (similar to capabilities in SmartThings). The devices in Apple HomeKit can be grouped by the specific type of services they can provide, thus, the implementation of an Apple HomeKit solution starts by activating a group of services previously defined. Services implement *characteristics* that define the interactions among services. Characteristics can be read–write (e.g., setting a new threshold value for temperature), write-only (e.g., specific commands for devices), and read-only (e.g., current temperature). Finally, *actions* are used to send commands to specific devices (e.g., close garage door), and *triggers* can be defined to execute actions at specific dates and times.

Microsoft IoT Core. IoT Core is an IoT platform developed by Microsoft that gears toward general IoT usage — the purpose being that IoT core can be implemented to fit any developer's needs [31]. IoT Core stem from the Universal Windows Platform (UWP) [36], which permits developers to make applications for any UWP compatible device (e.g., Xbox, PC, IoT devices, etc.). UWP allows for a developer to program for her *device* using *drivers*. From there, the developer grants the IoT device *permissions* to read, write, or communicate IoT data in the network. By connecting to Microsoft's Azure cloud platform, the user can implement a full IoT solution without the need for a hub-like device, allowing for easier implementation, scaling, and management.

FarmBeats. FarmBeats is an agricultural IoT platform also developed by Microsoft [32]. Although FarmBeats inherits from IoT Core, it develops a full independent IoT solution more tailored for smart agriculture. This platform boasts unique contributions to improve agriculture needs, such as: "long-term deployment, weather-aware devices, and novel inference techniques" [32], all designed to increase efficiency on a farm. Due to farms having unreliable Internet connections, FarmBeats has proprietary IoT gateways that function similarly to a hub. While offline, these IoT gateways can host localized data analysis and control

[3] We define as "extension" any handcrafted IoT app that can be used to control IoT devices. In general, these apps offer extended personalization capabilities to the user if compared with the official IoT app.

of IoT devices, and, when connected to the Internet, they upload the data to the Azure clouds for storing and long-term data analysis. FarmBeats is currently in public preview and is available in Azure Marketplace [37]. Compared to other platforms, the FarmBeats solution has additional challenges. In addition to the connectivity issues previously explained, FarmBeats developers also need to consider maintenance limitations due to the use of FarmBeats devices in remote farm areas. For instance, limitations on the access to reliable power sources and expected difficulties on the replacement of expired batteries from the devices impose the use of very low power communication protocols like BLE.

Amazon Web Services (AWS) IoT. AWS IoT is another general-purpose IoT platform designed to connect IoT devices to Amazon Web Services [33]. The critical focus of AWS IoT is to bridge IoT devices to Amazon Web Services so that data can be streamlined to the user. By using this platform, users can program their devices through specific *APIs* and *development kits*. From there, users may read and alter *device states* to manage live IoT data in her IoT solution. Optionally, IoT gateways or hubs may be installed to help bridge the connection or add additional functionality to the IoT solution. For instance, a smart homeowner can utilize the Alexa hub to control other devices via *Alexa skills*. These skills enable Amazon Alexa functionality and support through custom-built keywords and phrases. Taking advantage of its flexible architecture, more vendors are allowing the integration of their IoT devices with the Alexa hub. This way, the IoT user can benefit from more flexible and customized IoT solutions.

ThingWorx. ThingWorx is an industrial-based IoT platform designed to help automate businesses and implement remote monitoring of industrial processes [34]. The platform is built on a centralized application hosted on the cloud that developers can connect their industrial IoT devices to. A user implementing a ThingWorx solution can add support for their specific IoT devices through *APIs* and *SDKs*, which are uploaded to their ThingWorx application. From there, it is just a matter of adding devices in a plug-and-play fashion to get an IoT solution up and running. With many built-in features and industrial partners, ThingWorx promotes that its platform can be used in a variety of industries to help automate a company's work environment.

IBM Watson IoT Platform. Watson IoT is a general-purpose IoT platform developed by IBM [35]. While advertising for the industry, IBM notes that Watson IoT can be used for any IoT solution, including the smart home. Like other platforms, a user must add support for their device through specific *SDKs*. However, as the platform is integrated with IBM's Watson AI, IBM also promotes its machine-learning services to adopt into IoT applications and data analytics.

4. A novel evaluation framework for IoT Platforms

As discussed before, IoT platforms are unique in their applications, targeted domains, and functionality. Therefore, no two platforms can be compared easily with each other. For instance, one smart-home platform may focus on openness and the ability to rapidly add unsupported IoT devices while an industrial IoT platform would mainly focus on proprietary devices that function right out of the box and that are highly configurable. These two choices alone would have a broad impact on what resources (e.g., communication protocols, authentication mechanisms, and topology) are fundamental to consider while implementing IoT solutions. With this work, we aim to provide a comprehensive resource to IoT users (i.e., administrators, developers, and researchers) so they can make an informed decision on what IoT platforms to choose based on their specific needs. For this, platforms from different domains need to be considered. In this way, this research aims to impact a broader group of users. Finally, we note that we selected the platforms considered here based on their independence and support capabilities (see Section 3).

To compare fundamentally different IoT platforms, we define comparison criteria that directly impact three implementation pillars: (1) communications, (2) security, and (3) privacy. We believe that despite the desired application, IoT users want to guarantee reliable connectivity while preserving the CIA triage of security: confidentiality, integrity, and availability of the IoT data and services. Specifically, we consider the following seven comparison criteria: topology, programming languages and application development, device support, protocol support, event handling, security, and privacy. We define and briefly overview these criteria below:

Topology. This feature concerns with how the IoT solution handles the flow of information among the different sensors, devices, hub, and cloud servers. Since the majority of IoT devices themselves are too lightweight to process the amount of data they produce, there must be a dedicated instance at the center of an IoT solution capable of analyzing and storing the IoT data. There are three possible topologies: (1) centered around a local hub device, (2) with IoT devices communicating directly with a cloud-based server, and (3) a hybrid combination of the previous two. Analyzing how data flows is central to how one implements security policies for the IoT solution. For instance, security challenges and needs are different if the sensitive information is stored locally or is sent to a remote party over the Internet. In this criteria, we focus on how each of these topological choices may affect the performance and the security of the IoT solution.

Programming Languages and Application Development. The capacity of building tailored IoT apps is what gives the IoT solution its desired functionality. Like traditional operating systems, many programming languages can be used at the backbone of IoT. Some programming languages, such as Embedded C, are highly efficient at running on specialized devices while others, such as Java, are capable of working on practically any device. On the one hand, the IoT platform may allow full use of a programming language and give more control and resources to the developer to build specific apps tailored to her needs. On the other hand, the platform may provide a lightweight, stripped version of the language to ease the strain on the IoT devices' computing resources. In this criteria, we study what choices platforms make in their programming languages, how the platforms implement their APIs, and how these choices impact the IoT solutions' capacity to implement adequate connectivity, security, and privacy.

Third-party Support. By concept, IoT integrates numerous different types of devices that can operate in many ways. In fact, the majority of the current research work in IoT focus on implementing security solutions that target IoT platforms that support a high number of devices [7–9]. In addition, to increase usability, some IoT platforms allow the integration of third-party or handcrafted applications (i.e., extension) that offer an extended set of features either to control native or third-party devices to the user [38]. For that reason, we also study the support of an IoT platform to incorporate third-party apps and devices. In IoT, devices are only proprietary, allowing for easier installation for the user, or the platform provides tools to allow for third-party device support, giving the user a broader customization capacity in her IoT solution. Also, on the application support side, even though it is complicated to integrate applications that do not completely abide by similar programming languages and architectural approaches, freelance developers often offer software solutions that surpass "original" IoT apps. While analyzing these criteria, we study these two categories of providing third-party support, focusing on what would be the pros and cons or both in terms of security and privacy. For instance, providing support to third-party devices could open undesired doors to new security vulnerabilities as unknown compromised devices can be easily integrated into the solution [39–41].

Extended Protocol Support. IoT platforms offer a set of predefined APIs to implement specific protocols (e.g., `permission:entity-type:entity-id` is used to implement a REST call in Samsung

SmartThings). However, as the number of predefined APIs is restricted for most platforms, some specific protocols may not be implemented. As a consequence, limiting the set of available protocols necessarily limits the range of capabilities of the IoT solution. In this category, we survey how different IoT platforms allow for extended protocol support, so IoT users could plan for enhanced functionality in their solutions. In other words, we focus on how IoT platforms offer additional tools to the user (other than specific predefined APIs) to permit for the integration of additional protocols (e.g., communications, device discovery, public/private key exchange) that can be used to enhance an IoT solution. The broader the set of available protocols in the platform, the more tools the IoT user has to the support applications with higher technological demand. Among all the types of protocols, we further focus on those directly related to device connectivity. While devices are capable of operating individually, there must be a way to network them together to set up the IoT solution. In traditional computer networks, one could simply use communication protocols from the TCP/IP stack and assign all the devices IP addresses to enable communication. However, due mainly to specific challenging environments where IoT devices operate, many IoT devices are incapable of communicating in this way, and additional communication support must be added. A crucial element in IoT is the high connectivity required among devices and cloud-based servers. Indeed, high-speed connectivity is necessary in most cases to cope with real-time applications and the level of cooperation expected among IoT devices. In general, as in device diversity, IoT connectivity can be guaranteed in a proprietary fashion where the platform only allows devices that communicate over certain protocols. However, some platforms also provide tools that permit the integration of third-party communication protocols. As mentioned earlier, current IoT connectivity is mostly guaranteed via traditional network protocols and technologies like WiFi, Bluetooth Smart, and Device-to-Device (D2D) communications. However, specific IoT applications that require, for instance, the utilization of devices in a remote location are imposing the design and use of novel communication protocols that guarantee the required connectivity while imposing lower overhead in terms of power consumption and regular device maintenance. In this criteria, we first study what protocols are used in various IoT solutions, so the desired performance is achieved. Also, we highlight the impact of the approach used to select the communication protocols on security. First, the use of known vulnerable protocols may impact the overall security of the IoT solution. Second, the integration of third-party protocols may also open doors to new security vulnerabilities.

Event Handling. Trigger-action relationships in IoT settings are triggered by time, user input, or changes in the environment surrounding the devices. Developers call these changes in the environment *events*. The way an IoT platform handles the events may impact the performance and efficiency of the IoT solution. Effectively, events can be handled in a real-time approach, responding to the event as it occurs; or the IoT solution can periodically check for events and respond in precise intervals. In this criteria, we survey the methods used to handle events in an IoT solution and how this impacts the performance of the IoT solution. Also, security concerns can be raised from the analysis of trigger-action relationships. IoT researchers have studied these relationships to discover illegal states that directly impact security [42]. Finally, the analysis of trigger-action relationships permit the implementation of security solutions to discover malicious actors and compromised devices based on behavioral analysis [6].

Security. With the proliferation of IoT devices and services, securing them and improving the way they establish communications within local networks and with remote servers are among the highest priorities of the IoT user. We include additional criteria that evaluate how the IoT platforms evaluated here guarantee security via architectural design or through dedicated APIs to implement, for instance, encryption on the sensitive data that is sent out of the IoT applications. We study what mechanisms are provided by the platforms for administrators, developers, and researchers to secure their IoT solutions. These mechanisms support the essential security services of confidentiality, integrity, availability, and access control. For instance, we discuss authentication methods to secure who has access to the sensitive information handled by the IoT solution, the ability to implement roles and capabilities for access control, and other programmable security functionalities such as encryption of the sensitive information. We detail such mechanisms below.

- *Confidentiality*: We evaluate how IoT platforms are vulnerable to attacks impacting the confidentiality of users and services. Then, we also discuss the means that IoT platforms implement to protect or mitigate the effects of such attacks. We focus our discussion on how the IoT platforms guarantee confidentially by, for instance, implementing proper authentication mechanisms and encryption. Specifically, we evaluate how an IoT platform authenticates entities (i.e., users, devices, and applications). If an entity can perform malicious activities without needing to authenticate itself, it can cause severe damage to the IoT solution. Here, we survey how traditional authentication methods are adapted to IoT. Also, we evaluate potential new authentication methods that may be more tailored for IoT. In general, IoT devices are lightweight and lack the computing power of the average PC. However, fast communication between IoT devices is often required to exchange sensitive information that needs to be protected from passive and active malicious actors — especially when the devices operate critical infrastructure (i.e., the smart grid and healthcare equipment) and are in charge of real-time processes. Therefore, some encryption schemes (e.g., RSA) are not easy to implement on IoT solutions as the IoT devices are, for instance, incapable of operating at the required performance while still using encryption. This criterion discusses the encryption schemes that can be utilized by the IoT platforms to protect sensitive communication and stored data in the IoT.
- *Integrity*: Accurate data is key to support IoT functionality. The IoT paradigm follows the sensor-computation-actuator paradigm where devices react to commands sent from remote servers fed by and compute measurements received from sensors. Thus, the integrity of the IoT data defines the quality and effectiveness of the IoT services. The impact of data integrity is especially relevant in IoT solutions aimed at controlling and monitoring critical infrastructures (i.e., enterprise and industrial IoT) like the smart grid or modern water plants. We cover attacks that target the integrity of the IoT data. Then, we describe tools that IoT platforms may offer to the user to guarantee data integrity. Finally, we discuss potential security solutions that could be considered by IoT users to strengthen data integrity in IoT.
- *Availability*: The IoT solution is expected to offer "always-on" services. Disruptions within consumer IoT solutions may cause frustrations to the user. However, disruptions within the enterprise or industrial IoT may put critical services at serious risk. We discuss the main threats and cyberattacks, targeting availability in IoT. Then, we provide insights into what the different IoT platforms and users can do to mitigate such attacks.
- *Access Control*: Access control for IoT systems is similar to mobile phone permissions [43]. Each device in the IoT solution has capabilities (e.g., a light-switch can turn on or off), and the platform should have ways of allowing a device to use those capabilities. For this work, this criterion surveys the ways platforms provide users access to the device capabilities. Also, we evaluate if the platforms provide mechanisms to restrict access of certain users to specific capabilities. For instance, a homeowner may want to restrict access of an Airbnb guest to devices like the smart thermostat [16]. In summary, access control concerns on how the IoT platforms handle access control to both users and devices.

Privacy. Protecting the IoT solution from malicious actors or compromised devices does not keep sensitive information from being inadvertently or deliberately disclosed to third parties. Thus, we also include criteria to compare IoT platforms in terms of the protection they provide to the privacy of users. In this criterion, we uncover how privacy is affected through a combination of a platform's choice in previous criteria. Also, we highlight methods provided by platforms to specifically cater to privacy. This would include any mechanisms and functionalities utilized by the platforms to make user data private as well as policy agreements that define out how any platform's parent company adheres to privacy.

5. Evaluation of IoT Platforms

This section aims to evaluate the selected IoT platforms with communications-, security-, and privacy-impacting criteria. This evaluation includes analysis of topology, programming languages and application development support, IoT event handling, third-party support, extended protocol support, security, and privacy. We chose these criteria as they impact the security and connectivity capabilities of all the evaluated IoT platforms, regardless of how a platform implements specific IoT solutions, which is of a high priority to the user. We look into overall platform design patterns in IoT and point out the commonality between the different IoT platforms. After highlighting these trends, we then discuss the advantages and disadvantages of a particular IoT platform adopting specific practices. Finally, our goal is to provide the IoT users with an evaluation framework that allow them to make informed decisions regarding the design choices to implement their IoT solutions.

5.1. Topology

In this criteria, we are concerned with how the IoT platform handles the flow of information among sensors, devices, apps, and servers, as IoT devices and sensors are often too lightweight to analyze the amount of data they produce themselves, and therefore, they require a dedicated device (often a hub or cloud-based server at the center of an IoT solution) capable of analyzing and storing the IoT data. How data flows within a solution is key to how one defines further design rules. In this criteria, we focus on how each of these topological choices can affect the connectivity performance in an IoT solution and the security and privacy risks the different architectural approaches may arise. We find three possible design topologies in our analysis (Fig. 2): (1) centered around a local hub device, (2) with IoT devices communicating directly with a cloud-based server, and (3) as a hybrid implementation of the previous two approaches [44]. Below, we detail how these topologies function and identify benefits and detriments to their use.

Hub-based Topology. Generally, IoT solutions implementing this topology requires of a centralized entity (i.e., the hub) that is local to the IoT network. The hub serves as the central processing unit and the communication bridge between the connected IoT devices and the cloud-based services. OpenHAB is a great example of a fully hub-based topology. The hub has all the capabilities to collect, store, and process data from the IoT devices [28]. Fig. 2(a) showcases a generalized hub topology: all IoT devices in the IoT network connect to the centralized hub, which processes the incoming data and provides control to those devices. *Advantages:* In general, the advantages of using a hub for centralized communications are (1) localized data processing and storing and (2) offline data analysis. First, if desired, the user can keep her IoT data private and secure from third-party malicious actors since all the data is stored on her local implementation. Second, the IoT solution will continue to work even without an Internet connection – a connection is only needed for downloading updates or new features, which directly impacts the connectivity requirements. Additionally, if an Internet connection is required to support specific services, the hub-based cloud benefits from having a single entry point rather than multiple Internet connections from multiple devices, which increases security. *Disadvantages:* The drawback of a hub-based topology is cost and additional implementation effort. To implement this topology, the user needs to get a proprietary hub or a device capable of working as a hub before she can set up the IoT solution. For instance, in Apple Homekit, the user must get an Apple TV, iPad, or similar device to operate as a centralized hub within their IoT solution. Additionally, the performance capabilities of the hub directly impacts the performance of the solution. If the user does not use a high-end hub, she may experience lower performance overall as the hub will not have enough processing power to handle the data load. Also, hubs tend to have a high amount of initial configuration before it is ready to be used properly. In openHAB, the user must install and configure the hub before she is capable of installing the rest of the devices. Despite the advantages explained before, the use of a hub may also impact security. Even though the data may be considered more secure due to the limited Internet access, if a hub gets compromised, attackers may have access to all the IoT solution data at once, which may impose the use of precise security solutions [6].

Cloud-based Topology. It relies on Internet connectivity to manage the IoT solution remotely using cloud-based servers (Fig. 2(b)). In typical scenarios, the vendor that supports the IoT platform hosts a cloud service that allows the user to connect her IoT solution directly to it. The cloud service then handles all of the data exchange, processing, and storing from the different devices to execute the IoT application logic and to communicate with the IoT devices. *Advantages:* The main advantage of using a cloud-based topology is reduced implementation time and overhead. Typically, the IoT platform hosting the cloud service has a pre-configured package to install the service quickly. Then, all that is left to the user is to customize her installation based on her specific needs. The cloud also provides the advantage of faster processing speeds as high-end servers will be performing the processing on the IoT data rather than using a single resource-limited device (i.e., hub). The use of cloud-based configurations also facilitates the support of third-party devices and increased app customization. On the one hand, the use of new devices is not limited by the adoption of specific communication protocols that impose the implementation of local networks. Instead, only reliable Internet connection capabilities are required for new devices to be included in the IoT solution. On the other hand, IoT developers or researchers can implement their own cloud services to increase app customization or to integrate novel security and privacy mechanisms [7,8]. *Disadvantages:* The major disadvantage of using a cloud service comes as increased connectivity requirements if compared to the hub-based approach. Using a cloud-based service requires an "always-on" Internet connection since the IoT solution cannot properly function in offline mode. Also, the user's IoT data has added security risks of being stolen or compromised either during the communication process or as soon as it is stored somewhere out of the local solution, which impacts security and privacy. As all the devices included in the IoT solution continuously exchange data with the cloud, new security measurements are offered by the platform to protect the information. For instance, the adoption of secure HTTP protocols like HTTPS that encrypt the information may be required to protect the sensitive information from, for instance, eavesdroppers. Finally, cloud-based architectures require trust from the user that the IoT platform can adequately protect the solution against external attackers or even passive observers. However, recent research works have demonstrated that malicious or handcrafted IoT applications may not correctly protect the IoT communications or even are capable of sending the data to malicious servers [8,9].

Hybrid Topology. The most common IoT topology adopts a hybrid architecture that combines the two previous approaches. In this case, the

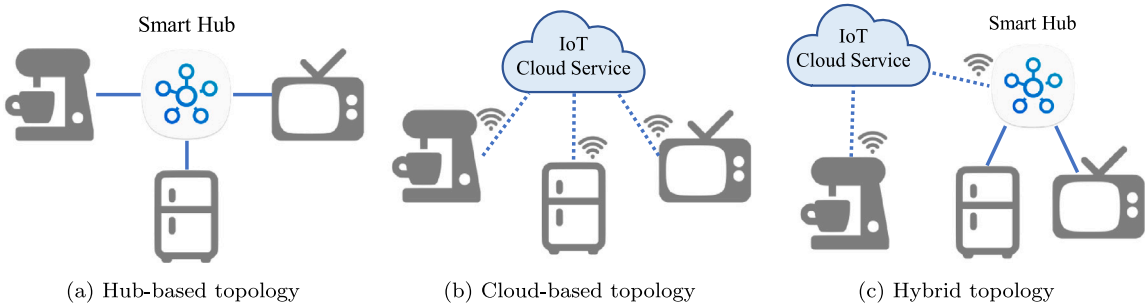


Fig. 2. Most common IoT topological design choices. Every approach differently impact the connectivity capabilities of an IoT solution. Also, they arise diverse security concerns to the IoT user and the system.

Table 2
Topology comparison among the different IoT platforms. Notice that, for all the platforms, either hub-based, cloud-based or both topologies are required. That is, they all have the capability of implementing hybrid approaches. (●= Required, ◐= Optional, ○= Not Used)

Topology comparison		
Platform	hub	Cloud
openHAB	●	◐
SmartThings	●	●
HomeKit	◐	●
Windows IoT Core	◐	●
FarmBeats	●	●
AWS IoT	◐	●
ThingWorx	◐	●
Watson IoT	◐	●

IoT solution includes a centralized hub that "manages" and "monitors" devices in the solution, but also exchanges data and communicates with a cloud service. Also, individual devices are capable of connecting with the remote server directly as shown in Fig. 2(c). *Advantages:* There are distinct advantages in the use of hybrid approaches to implement IoT solutions. First, the IoT solution does not lose performance in the absence of a reliable Internet connection as hybrid solutions can be used in both online and offline modes. Second, when possible and required, the data processing and storing workload can be shared by both the hub and the cloud. For instance, for services that are more time-sensitive and require the interaction of various devices in the network, the hub can be used as the primary managing entity rather than servers that add extra communication latency due to the use of secure communication protocols (e.g., HTTPS). *Disadvantages:* The main drawback of the hybrid architectures concerns with the security and privacy of the information. As explained before, hub-based approached are risky as the sensitive data from the complete solution can be at risk if one single device, the hub, gets compromised. This security challenge is partially solved in cloud-based solutions as devices can communicate with the cloud-based servers individually, so the risk of compromising the whole solution when single devices are under attack is minimized. However, privacy concerns arise from the idea of sharing the IoT data with third-party providers in the cloud. In hybrid solutions, IoT platforms must provide the necessary security features to protect the sensitive information in both local devices and the cloud. In the cases where current solutions cannot support protecting the security and the privacy of the information, the IoT administrators or developers must be able to design and implement third-part solutions that properly secure the systems.

5.1.1. Comparative analysis and summary

Table 2 details the main topology approaches that the different IoT platforms surveyed use. Notice that, for all the platforms, at least one of the topologies hub-based or cloud-based is required, meaning that they all have the capability of implementing hybrid solutions. Due to a large amount of processing required to analyze streams of IoT data, most of the IoT solutions are built based on a cloud-based approach. Then, to compensate for the fact that some IoT devices are incapable of an Internet connection, different platforms provide the option to add a hub to the deployment, acting as a connection bridge between the IoT devices and the cloud servers. Some platforms – like Microsoft FarmBeats – use both a hub and remote servers (which we refer to as a hybrid topology) for their implementation strategy (Fig. 2(c)). As explained before, platforms like FarmBeats that focus on agricultural applications may need to use both a hub and a cloud server since the average farm has unreliable broadband connections [32]. Thus, real-time data analysis is performed by the IoT hub to allow for offline communications, as a farm will regularly have periods of no Internet connectivity. When available, long-term data is then sent to the cloud for further processing and storing. For instance, the FarmBeats hub performs daily humidity and temperature analysis, and the remote services can use this data to calculate seasonal patterns such as when to perform crop-rotations. In summary, whichever topology choice is used, centralized communication is the bottleneck of the IoT solution. Either the user must invest in a well-performing hub or a reliable Internet connection to increase the efficiency of her IoT solution. Also, the IoT user must consider the security and privacy challenges that every topology has, especially in hybrid cases where both local and remote threats need to be considered.

5.2. Programming languages and application development

IoT platforms use different programming languages to help automate an IoT solution. By using programming resources, the IoT platform can analyze IoT data and control the devices. In this criteria, we discuss the programming languages that the different IoT platforms use to provide support and resources to the IoT solution. Also, we analyze how the different choices may benefit or affect the IoT solutions in terms of overall performance, flexibility to add or create new features, and the ability to protect the sensitive information and the privacy of users.

Programming Languages. The development of the IoT platform may use different programming languages (e.g., Groovy, C, or Java). Such diversity, while desirable for implementing IoT solutions for specific

Table 3
Programming languages supported by the different IoT platforms surveyed in this work. We also summarize the platforms' capability of implementing sandboxing and DSL. (●= Required, ◐= Optional, ○= Not Used).

IoT Platform programming language comparison			
Platform	Supported languages	Sandboxing	DSL support
openHAB	Java	◐	●
SmartThings	Groovy	●	●
HomeKit	Swift	○	○
Windows IoT	C++, C#, Python	○	○
FarmBeats	C++, C#, Python	○	○
AWS IoT	Embedded C, JavaScript, Java, Python, Swift, C++	○	●
ThingWorx	C, Java, C#, Swift	●	●
Watson IoT	Python, Node.js, Java, C#, Embedded C, mBed C++	○	○

applications, will impact aspects of the final implementation (e.g., runtime, overhead). In general, every IoT platform provides the specific APIs used to develop and implement the final IoT solution. Nevertheless, the use of different programming languages can make the solution vulnerable to specific language-related challenges. *Advantages:* Diversity in IoT may be beneficial for implementing IoT-specific solutions. For instance, IoT solutions in agriculture have different characteristics from a smart home environment (e.g., a smart home would require close to real-time analysis of the IoT data to keep the home secure whereas agriculture would require periodic, more long-term data analysis to analyze crop patterns). Also, specific APIs may facilitate the addition of new features, increased customization capabilities, and the support of third-party devices, all desired in smart home solutions. For this, different programming languages can provide the required diversity of tools for efficiently-tailored development of IoT solutions. *Disadvantages:* Programming language diversity causes the apparent detriment of niche specialties. An IoT application developer skilled in Java would be well-suited to develop applications for openHAB, but the same developer would have difficulties developing for Windows IoT Core, which does not use Java. Additionally, any application that an IoT developer makes for one programming platform will not necessarily work for another due to compiler incompatibility, different API structure, or different data type definitions, leading to the need for extended development time for cross-platform implementations. Finally, programming diversity also carries additional security challenges. First, IoT users need to consider the security vulnerabilities of the specific programming language. For instance, for applications developed for ThingsWorx and Watson IoT, cyber attackers can take advantage of unsafe C functions to leak information through memory overflow type of attacks. Also, in an application developed for SmartThings, attackers can use Groovy language-specific features like dynamic method invocation to change the behavior of the IoT apps.

Sandboxed Environments. With IoT devices, it is often not necessary to include the entire functionality of a programming language (e.g., threading, dynamic memory allocation). One reason for doing this is to guarantee performance in resource-limited environments as only the required functionality for an app to work properly is allowed. Also, limiting the type of methods and classes implemented in an IoT solution provide some added security to the development process as unsafe functions can be blocked. To allow this, IoT platforms often *sandbox* the programming language in their platform. Sandboxing limits certain functionality of a programming language to only those deemed necessary for the implementation. For instance, SmartThings implements the Groovy programming language in a sandboxed-environment [29], which limits a developer to easily open a file or declare a global variable that can be used later to leak data related to device's events [9]. *Advantages:* Implementing a programming language in a sandboxed environment is beneficial in two ways. First, there is less overhead related to unnecessary processes for the IoT devices, allowing them to run more efficiently. Second, with security in mind, there are fewer attack vectors available to malicious actors. For instance, in the Samsung SmartThings platform, a developer can utilize platform-available functions that properly encrypt and protect the information before sending it to remote servers. However, the same developer does not have access to other encryption mechanisms to encrypt and leak sensitive data to unauthorized recipients using a back door, hiding her actions from currently available privacy-analysis tools. Finally, sandboxing permits for the implementation of third-party security solutions in IoT, that provide additional support the security and privacy efforts offered by current IoT platforms [45]. *Disadvantages:* Sandboxing a programming language takes time to implement correctly, requiring additional configuration by the IoT platform before their product can be available to the market. Additionally, the limitations of a sandboxed-environment may be a deterrent to developers of the platform as they must necessarily learn what can and cannot work while writing code for the platform. Finally, despite this seems as an advantage if analyzed from a different perspective, limiting the types of software methods and classes available to the developer also restricts the customization capabilities an app can offer to the IoT users.

Domain Specific Languages. They are primarily specialized programming languages that are designed for specific applications [46]. A typical example that can be found in database management is SQL [47]. SQL is designed to manage and operate on large datasets. While SQL cannot perform all the functions of a general programming language, it can perform all the functions required to maintain normal operations of a database. Following the same principle, an IoT platform may adopt a DSL to allow the development of lightweight IoT applications and scripts. The IoT DSL would only need to be capable of supporting simple yet powerful scripts used to manage and control events and actions in IoT devices. For example, openHAB uses Xtext, which is a framework that allows users to create custom DSLs based on their own needs [28]. The use of Xtext allows an IoT user to build abstractions (i.e., concepts such as switches, buttons, a time of day) to ease the setup of the IoT solution in the home. *Advantages:* Similar to sandboxing, adding a DSL to an IoT solution simplifies the code efforts from the user. The DSL can be written in a way that permits the abstraction of complex programming concepts into simple operations or "things". For instance, as shown in Listing [28], openHAB uses the concept of "rule" to convert complex trigger-action relationships among devices within the IoT solution into simple behaviors (e.g., `when` *motion_ON* `then` *light_ON*). *Disadvantages:* While sandboxing aims to reduce the computational processing imposed on IoT devices by unnecessary processes from apps, using DSLs to an IoT solution may add overhead. The abstracted programming concepts need to be translated back into appropriate machine code for an IoT device to run the code, which increases the complexity of the operations performed at compile time. Another major disadvantage of DSL is that, by "hiding" real software operations behind simpler abstractions, it also makes the security analysis of IoT apps more difficult. For instance, dynamic security tools used to analyze IoT apps may not be able to infer the app's intent or its behavior without previously analyzing and understanding the DSL script derived from the app.

5.2.1. Comparative analysis and summary

We find that the programming languages and their respective implementation choices vary the most among our chosen criteria. Table 3 summarizes the preferred programming language for every IoT platform. From left to right, we list the programming languages that are supported by the different IoT platforms, whether the software resources are implemented in a sandboxed environment, and if the specific platform supports the construction of a DSL in its programming environment. One can observe that for some platforms, different programming languages are possible.

Listing 1: OpenHAB Sample Code

```
rule <RULE_NAME>

when
    <TRIGGER>or<TRIGGER2>
then
    <EVENT_HANDLE_BLOCK>
end
```

Listing 2: SmartThings Sample Code

```
def eventHandler(event){
    def data=parseJason(event.data)
    def trigger(data){
        TRIGGER BLOCK }
    log.debug "event data:${data}"
}
```

Listing 3: HomeKit Sample Code

```
init(
    name:EventHandle,
    events:[HMevent],
    end:[HMevent],
    recurrences:[DateComponents],
    predicate:NSPredicate
)
```

Listing 4: IoT Core Sample Code

```
//Start capturing events
//and save them to an ETL file
xperf -start <Session Name>-f
    <ETL File> -on <GUID>
//Stop capturing events
//with the specified name
xperf -stop <Session Name>
```

Listing 5: AWS IoT Sample Code

```
"Version":"2012-10-17",
"Statement":[{
    "Effect":"Allow"
    "Action":["iot:Subscribe",
        "iot:Recieve"],
    "Resource":[".../aws/events/"]
}]
```

Listing 6: ThingWorx Sample Code

```
var trigger =this.domElement
    trigger.triggerHandler("Event")
    .
    .
    .
    .
```

Listing 7: Watson IoT Sample Code

```
"$schema":"json-schema.org",
"type":"object",
"title":"Event",
"description":"Event trigger",
"properties":{"Event":{"type":"object",
    "description":"Device event",
    "$logicalInterfaceRef":"Event"}}
```

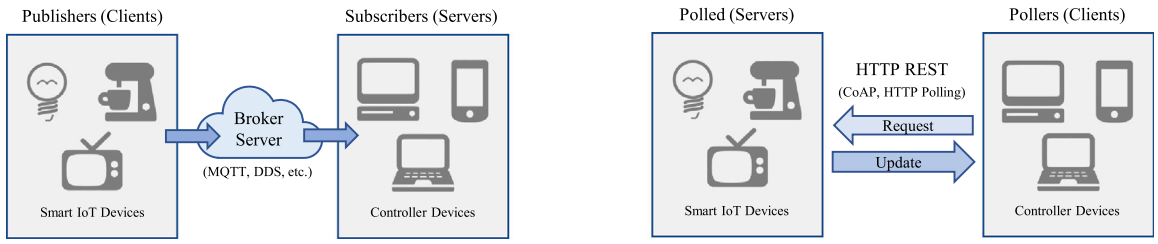
Additionally, in Listings [28] to 7, we provide an event trigger example to showcase the differences in program development for each of the evaluated IoT platforms. If only the programming language that the platform implements is considered, choosing a platform over another one mainly accounts for the IoT user's programming skills. More savvy programmers in specific languages could move faster across the development process. Also, the selection of more secure functions to implement the APIs must be guaranteed by both the IoT platforms (via sandboxing) and the developer (via more specialized programming skills), so the selection of more secure languages (when possible) like Java can help the programmer to remove some of this burden from the IoT solution implementation process. Finally, selecting platforms that support the use of DSL may help to reduce the complexity of the implementation; however, that may be possible with the compromise of security.

5.3. Event handling

An *Event* is a colloquial term used in IoT to describe the reactions of IoT devices to real-time physical changes in the surrounding environments (e.g., temperature sensors responding to increased heat). The method in which IoT platforms handle events is crucial for handling application logic and deployment requirements. Event handling in IoT is performed at the application level; thus, several protocols are offered to do so; however, all these protocols seem to work using two specific approaches (Fig. 3): (1) a *publisher/subscriber* model and (2) a *polling* model. In this subsection, we give details on how both methodologies function, including the most-common application-level protocols used, and discuss how the use of either method impacts the IoT solution.

Publisher/Subscriber Approach. The most common method for event handling in IoT platforms is the publisher/subscriber approach. In this method, an IoT device within the solution (i.e., the publisher) acting as a client streams its data in real-time to specific controller devices (i.e., the subscribers), who are in charge of providing setting measurements to the devices. In between publisher and subscribers, a *broker server* acts as a bridge (Fig. 3(a)). A simple example of a publisher/subscriber model is an air-conditioning system in a smart house. Assume that in a smart home there are three types of devices installed: temperature sensors, the smartphone that has the controller IoT app installed on it, and the controlling unit that manages when the air-conditioning is being used. Whenever a temperature sensor detects a temperature change in the room, it publishes data to the listening AC unit. The AC unit reads this change and reacts accordingly – such as turning off the AC in that sensor's room – based on the setting values received from the controller app. There are several different application-level protocols used to implement publish/subscriber solutions. In the following, we are detailing some of the most popular ones.

- *Message Query Telemetry Transport (MQTT)*: In an MQTT setup, a MQTT server – typically called an MQTT broker – runs on the IoT solution [48]. From there, a "publisher" and a "subscriber" link themselves to this broker under a common identifier. In an IoT solution, the IoT devices act as the publisher, and IoT hubs or control devices are the subscribers. Whenever the publishers have new data to record, they publish the data to the broker. The broker then flags that it has new publisher data, and the subscriber reads the data. From there, the subscriber can analyze the data and react accordingly.
- *Advanced Message Queuing Protocol (AMQP)*: This is an open standard application-level protocol that guarantees message delivery in three modes: (1) at-most-once, in which the delivery of the message is not guaranteed; (2) at-least-once, in which the protocol makes the best effort to deliver the message at least one time;(3) and exactly-once, in which the message reaches the destination only one time [49,50]. AMQP is suitable for heterogeneous IoT solutions where devices from different vendors co-exist. Also, it does not require a reliable Internet connection at all times to function as the AMQP broker can be used in modes where the message is delivered as soon as a connection between the devices is possible.
- *Extensible Messaging and Presence Protocol (XMPP)*: This open protocol is based on Extensible Markup Language (XML) and is suitable for applications that require real-time analysis [51–53]. Even though XMPP supports both publisher/subscriber and polling modes, we include it in the first group as bidirectional communications is not a requirement for this protocol to properly work.



(a) Publish/subscribe mode using a broker server

(b) Polling mode using REST protocol

Fig. 3. Examples of the two event handling methods used in IoT: (a) publish/subscribe mode via MQTT and (b) polling mode via REST protocol.

One major advantage of XMPP is that it implements authentication and encryption capabilities through Simple Authentication and Security Layer (SASL) and Transport Layer Security, respectively. As for drawbacks, this technology has higher overhead compared to other similar application-layer protocols, and it is difficult to guarantee that messages are always delivered with the required quality of service.

- *Data Distribution Service (DDS)*: This protocol was originally designed to implement machine-to-machine (M2M) communications in real-time IoT applications [54–56]. Different than XMPP, this protocol supports quality of service and always guarantees the reception of the messages. However, it does not need a centralized broker (as in MQTT) to manage the message delivering process. In fact, it used multi-cast among all the devices included in the solution as the main delivering mechanism. In DDS, the publisher and subscriber roles vary with the specific applications. For instance, the motion sensor (publisher) directly broadcasts when movement is sensed inside the room so other devices, as the smart like or the smart camera (subscribers), can take action proper action. DDS is useful when new devices are continuously added to the IoT solution as it uses a plug-and-play approach. That is, DDS automatically assigns publisher and subscriber roles to the devices within the solution, so the type of data they use and its flow direction are known. This allows for DDS to be both scalable and easy to implement. Finally, DDS supports Datagram Transport Layer Security (DTLS) and Secure Socket Layer for secure communications.
- *Simple Text Oriented Message Protocol (STOMP)*: This is a simple and asynchronous protocol that uses intermediate servers to permit exchanging messages between different entities, implementing a subscriber/publisher architecture [57]. This protocol is supported in several programming languages through specific APIs. However, such APIs normally implement a very limited set of messaging operations, which makes STOMP very suitable for the IoT resource-limited devices. Also, this protocol is supported by the Transmission Control Protocol (TCP) communications, which make it very reliable.
- *ZeroMQ*: This protocol can implement publisher/subscriber architectures with or without using a dedicated broker server in between. The second mode provides a simpler implementation with reduced latency. ZeroMQ is supported in numerous programming languages. Also, it is very lightweight and permits the many-to-many high-speed asynchronous messaging between different endpoints and using different communication modes like in-process, inter-process, TCP, User Datagram Protocol (UDP), and mulicast [58,59]. Finally, ZeroMQ uses CurveZMQ to guarantee security via encryption of the data and also authentication capabilities.

Advantages: In the publisher/subscriber model, the IoT devices (publisher or clients) perform a limited set of operations to guarantee proper data exchange with the subscriber or controller devices: (1) read for sensory changes and (2) publish any changes — making this process very lightweight for the resource-limited IoT devices. Additionally, using publisher/subscriber models permits the IoT solution to react faster, thus support applications that require real-time analysis. This is beneficial to environments such as the IoT users expect their devices to respond as they are being used. *Disadvantages:* The obvious downside of publisher/subscriber is that, in most cases, an intermediate broker server is needed. Thus, either the IoT devices must always be connected to the Internet to publish to the IoT platform's dedicated broker, or the user has to go through additional steps to set up her offline MQTT broker. Also, every different application-layer protocol that implements the publisher/subscriber architecture handles security differently, which creates more complexity for the user implementing the IoT solution. For instance, while DDS uses general-purpose protocols like DTLS and SSL to implement security, ZeroMQ implements its security protocol, CurveZMQ [60], which uses CurveCP [61] and NaCl [62] to guarantee encryption of the data and authentication capabilities.

Polling. The second method for event handling supported by IoT is polling, which uses a client/server approach. Here, there exists a poller device (a controller device acting as a client) that periodically connects to the IoT devices (acting as servers) to request for some data. Notice that while in the publisher/subscriber method the IoT devices act as clients and controller devices as servers, their roles are swapped during polling. For instance, the controller devices or pollers periodically asks the IoT devices for their current states, and the devices respond with the most recent readings from their sensors. While not in real-time, this method allows for data to be read in precise intervals. We can use the air-conditioning system again as an example. The AC unit (acting as a poller client) periodically asks sensors (the polled servers) throughout the house what the current temperature in their rooms is. The sensors respond with their current temperature readings. From there, the AC unit can again react accordingly based on individual room temperatures. Representational State Transfer (REST) [63] is the lower-layer protocol that most commonly implements and supports application-layer protocols for polling in IoT (Fig. 3(b)). When using REST, there is no need for a broker server acting as a middleware supporting the communication between IoT devices and controller devices. Instead, two devices can communicate directly (often over an HTTP connection) in a typical client/server fashion [64]. The client (can be the hub, the cloud-based server, another IoT device, or a controller device (smartphone or laptop running the controller application)) requests data from the server (IoT device) who responds with the data currently stored in its system. In the following, we are discussing some application-layer protocol that implements polling-type event handling in IoT.

- *Constrained Application Protocol (CoAP)*: This is a polling protocol specially designed and tailored for resource-limited IoT devices [65–68]. With IoT devices in mind, the protocol allows for exchanging messages between devices in the same network or devices located in different networks connected via the Internet with very low overhead. CoAP is based on the REST architecture mentioned before and supports multicasting via UDP protocol. Specifically, CoAP implements four different types of messages: (1) confirmable, non-confirmable, reset, and acknowledgment.

Finally, some efforts have been made to integrate security into CoAP. However, these efforts still under development mainly because security integration affects the lightweight nature of the protocol. Currently, as in DDS, CoAP protocol relays on lower-layer protocols like DTSL and IPSec.

- *HTTP Polling*: The Hyper Text Transfer (HTTP) protocol is the underlying communication protocol of the World Wide Web (WWW). Created in 1997, HTTP acts as a request–response protocol in the client–server computing model. Controller devices use HTTP polling to request information from IoT devices. HTTP polling can be either short or long. In HTTP Short Polling, an IoT device (i.e., server) may receive and handle multiple requests from the controller devices (i.e., clients). These requests are processed in the order they are being received, but they are kept open only for a short period of time, yielding a higher network traffic, but lower utilization of the IoT devices' resources. In HTTP Long Polling, only one polling request is handled at any time, and the requests are kept open until the server responds back with the requested information [69–72].

Advantages: Since polling directly implements device-to-device communications, it makes the communication setup more straightforward for the user as no intermediate broker server is needed. Additionally, as polling often implements HTTP REST communication principles, many devices with web capabilities would be able to adopt this schema by merely connecting through an HTTP port, which makes this communication mode very suitable for IoT devices. *Disadvantages:* On the other hand, polling requires additional computation power from individual resource-constrained IoT devices. With polling, the IoT devices must continuously listen, so they do not miss periodical poll requests from the poller. For some IoT devices, adding such an additional task can severely impact the device performance as its computation power is often the bare-minimum to perform its job. Also, the user must consider the particularities of every protocol to guarantee the security of the IoT solution. As explained earlier, some application-layer protocols implement their own security mechanisms with dedicated solutions like CurveMZQ; however, the two polling protocols rely on well-known security features provided by lower-layer communication protocols like DTSL and HTML5 which may limit the security configuration capabilities for the user.

5.3.1. Comparative analysis and summary

In our analysis, we find that all of the platforms that we evaluated used some version of the publisher/subscriber method to handle events. Typically, the platforms implement an MQTT broker that IoT devices use to publish their data. Some platforms, such as Amazon AWS, added optional support for polling using the client/server approach [33]. Table 4 summarizes the way each platform handles the exchange of event data between different devices within the same solution or the cloud. Typically, the platforms always require the implementation of publish/subscribe methods to handle events on the IoT solution. In some instances, such as AWS and ThingWorx, the frameworks have support for either publish/subscribe or polling, and it becomes up to the user to choose what works best for their implementation based on the specific application-layer platform used.

5.4. Third-party support

As with any computing paradigm, IoT platforms may allow developers to add their own functionality to the individual IoT solution. Third-party development can come in the means of developing tailored IoT applications, adding support for new communication protocols, or even introducing their own unique IoT devices to the solution. With this criterion, we study the benefits/detriments to an IoT platform introducing support for third-party development.

IoT Applications and their Extensions. Third-party support often comes in the form of new "tailored" applications for the IoT solution, but can also be packaged as extensions or complete overhauls of the platform. For instance, HomeKit allows applications from the Apple Store to control aspects of a user's HomeKit solution. On the other hand, since openHAB is open-source, a developer can easily create extensions of openHAB to add or completely alter the core functionality of openHAB. While the level of control differs for the platform, the concept of outside development remains the same: the platform offers tools (e.g., SDKs, APIs) for other developers to add their own functionality to the solution. *Advantages:* Allowing third parties to add functionality to an IoT platform is useful for a user to customize her IoT solution. Instead of going through the very time-consuming process of designing new and necessary functionality herself, an IoT user can install the tools someone else created to broaden the use of her own IoT solution. A secondary benefit of third-party app support relays on the paradigm that more people working on a problem may usually lead to quicker innovation. IoT is still a new under-development concept, and more people rooting for innovation may bring better and more suitable working IoT solutions. *Disadvantages:* The most critical aspect around adding third-party app support to an IoT solution concerns with security and privacy. Because third-party apps do not always go through a detailed and comprehensive vetting process, IoT platforms cannot be held accountable for malicious or unauthorized activities within the app that may compromise the sensitive data and the privacy of users. Previous research works have demonstrated that handcrafted apps (i.e., extensions) may contain malicious code to leak information to unauthorized recipients [7–9]. Also, adding third-party support is often time-consuming and difficult on the side of developers of the IoT platform. Additionally, developers of the platform often have to learn what is effectively entirely new APIs to develop new applications in the specific IoT platform.

Supporting Third-party Devices. The current IoT market is mostly focused on the "ease of living" automation. However, providing extended usability in IoT requires a high diversity of devices and sensors that not all platforms are in capacity to assume the increased costs and technical overhead. To overcome these challenges, some IoT platforms allow the support of third-party devices that permit the implementation of missing new functionality within the solution. The capability of adding new external devices is not symmetrical in all the platforms. For instance, proprietary platforms like SmartThings and HomeKit boast numbers of over 390 [29] and 280 [30] supported devices, respectively. With the more open platforms like openHAB and Windows IoT Core, it is harder to determine an exact number of native supported devices, however, the number of total devices would be similar as third-party devices can be directly added to the total population of supported devices [28,31]. On the other hand, IoT platforms focused on business sectors would likely require the support of third-party devices in their implementations as any given field may require dedicated device functionalities. Therefore, it is observed that industrial IoT platforms have considerably more IoT device support as off-the-shelf devices (e.g., cameras, drones, and medical devices) can be given support in an IoT platform through the use of developer SDKs and APIs. *Advantages:* Similar to the case of application support, the feature of adding third-party device support brings enhanced functionality to the IoT solution. Guaranteeing such a level of functionality makes the IoT platform more attractive to the IoT users. Also, in most cases, providing software solutions like dedicated APIs to support external devices is substantially simpler than dealing with the unreasonable extra burden on behalf of the IoT platform (e.g., through dedicated protocols, hardware, and software configurations) to provide support to an extended number of devices. Therefore, it is of benefit to the platform to provide software to allow users of an IoT solution to integrate their own devices. *Disadvantages:* The larger diversity of IoT devices as a result of third-party support becomes its own setback when it comes to usability. Providing generalized functionality for a large variety of IoT devices becomes a sizable measure that may impact the

Table 4
Comparison of event handling approaches used by the different IoT platforms surveyed. One can observe that most platforms require the implementation of publisher/subscriber approach (●= Required, ◐= Optional, ○= Not Used).

IoT Platform event handling comparison		
Platform	Publisher/Subscriber	Polling
openHAB	●	◐
SmartThings	●	◐
HomeKit	●	○
Windows IoT Core	●	◐
FarmBeats	●	◐
AWS IoT	◐	◐
ThingWorx	◐	◐
Watson IoT	●	◐

added functionality. For instance, in most cases, users desire plug-and-play usability that can only be obtained with out-of-the-box solutions provided by the IoT platforms. However, adding third-party devices to a specific solution may require more advanced programming skills to integrate the dedicated APIs with the new hardware. In the end, the implementation burden is somehow removed from the platform and carried out to the end-user. It, therefore, becomes desirable for IoT platforms to establish partnerships with device manufacturers who will directly program the desired functionality to create out-of-the-box-type of solutions. Another distinct disadvantage derived from the support of third-party devices relates to the security of the solution. As with external apps, integrating third-party devices brings security concerns as comprised devices are known to impact the behavior of the IoT solutions [14]. Also, compromised devices have been linked to threats that poison sensitive measurements within the IoT solution or leak information to malicious external actors [13,73].

5.4.1. Comparative analysis and summary

The comparative analysis among the different IoT platforms regarding third-party support of external devices and apps is two-fold. On the one side, adding support of both external applications and devices brings a very desirable increased functionality to the IoT solutions. On the other hand, security concerns are raised as a result of incorporating new devices and apps that do not have the complete support of the IoT platform. As detailed in Table 5, only three platforms (openHAB, Windows IoT, and AWS IoT) are capable of integrating both external devices and apps. For the other cases, the platform either offer support to apps, devices or does not offer third-party support at all. Recall that the extended capability of third-party-enabled solutions is achieved at the cost of increased implementation complexity for the user. First, IoT users must select the platform that offers the necessary support for the specific application via native plug-and-play apps and devices. If third-party integration is required, users must look for platforms that offer enhanced security capabilities to protect against compromised devices and apps, and simpler APIs for faster integration.

5.5. Extended protocol support

A unique feature in IoT is the full range of protocols used by IoT devices [12] to guarantee the necessary usability. For any device, there can be a separate protocol for any specific functionality. Different protocols can be used to guarantee the required connectivity among IoT devices and servers (e.g., WiFi, Bluetooth, BLE, NFC, ZigBee, LTE), to send and receive data in unique formats to support specific applications (e.g., REST, MQTT, TCP/UDP), and to implement network integration (e.g., IPv4/6, 6LowPAN). Also, other protocols are used to increase the flexibility and functionality of the IoT solution (e.g., UPnP for device discovery, X.509 for public-key infrastructure) [74]. In most cases, IoT reuses protocols that were designed to support devices and apps from different ecosystems (e.g., traditional computing systems); however, there is current work undergoing to bridge the gap in IoT diversity through standardizing an IoT protocol suite [75]. Until it becomes commonplace, manufacturers of IoT devices are responsible for adopting the protocols that best fit their individual needs, resulting in an increased diversity of protocols used to cope with the required functionality in an IoT solution. It is challenging for the IoT platforms to account for all of the available protocols that can be possibly used to implement the IoT solution. Thus, the platforms have no choice to either (1) limit the number of supported protocols or (2) *provide the necessary support so IoT users can implement the protocols themselves within the solution*. The main limitation of leaving protocol implementation up to the users is that they would also be required to instrument the protocol security themselves, which can result in inadequate or limited security, or no security being implemented at all. Finally, even with the vast amount of protocols in use, some platforms introduce novel communication methods to overcome barriers where traditional communication standards are not feasible in the platform's field. For instance, most remote agriculture farms do not have the required coverage for a reliable Internet connection. Thus, a communication protocol that implements narrow-band connections would be more suitable than other protocols that allow for communications with higher throughput. As an example, FarmBeats introduced a way for their IoT devices to communicate over unused Television channel frequencies [32], allowing the IoT solution based on this platform to have enough bandwidth for reliable communication among field devices.

Communication Protocols. Since connectivity is one of the main requirements for IoT, we present some of the most popular communication protocols used in IoT. We highlight features, essential modes of operation, and specific applications of every protocol. Then, we summarize some other technical characteristics in Table 6.

- *Near-field Communication (NFC)*: This an open-source protocol that supports short-range low-speed wireless communications between IoT devices in proximity [76]. NFC requires devices interacting in two different modes: active and passive. The device in passive mode can only be used to send data or being read from an operation that is performed without the need for a power source connected to the passive device. For instance, the typical way of implementing passive NFC is through the use of physical tags with embedded information. Examples of embedded information could be the contact list of users, payment information, and configuration settings. On the other hand, devices in active mode can send and receive data. Thus, devices in active mode require the use of a power source to read and write from/to tags in passive devices or to directly connect and exchange data with other compatible devices.
- *Bluetooth*: Different from NFC, Bluetooth is a standardized protocol used to send and receive data between two paired and connected devices [77,78]. This protocol shares the same band with ZigBee and WiFi to transmit data in a master/slave approach. In this model, a master Bluetooth device can be connected to several slaves (i.e., up to eight) to form what is known as *Bluetooth piconet*. The master device coordinates the communication process within the piconet and can request or send data

Table 5
Capability of the surveyed IoT platforms to support third-party applications and devices. Some platforms offer dedicated SDKs and APIs to permit the integration of external devices and apps to increase the functionality of the IoT solution (●= Required, ◐= Optional, ○= Not Used).

IoT Platform third-party support		
Platform	Applications	Devices
openHAB	●	●
SmartThings	●	○
HomeKit	●	○
Windows IoT	●	●
FarmBeats	○	○
AWS IoT	●	●
ThingWorx	○	●
Watson IoT	○	●

Table 6
Overview of some of the most popular communication protocols used or proposed for IoT.

IoT Platforms communication protocols overview			
Communication protocol	Rate	Transmit range	Frequency band
NFC	424 kb/s	20 cm	13.56 MHz
Bluetooth	2.1 Mb/s	100 m	2.4 GHz
Bluetooth Low Energy (BLE)	0.27 Mb/s	100 m	2.4 GHz
WiFi (802.11ac)	1.3G b/s	50–70 m	5 GHz (2.4 GHz in prior versions)
ZigBee (802.15.4)	20–250 kb/s	10–20 m	2.4 GHz, 900 MHz, and 868 MHz
Z-Wave	40 kb/s	100 m	800 MHz–900 MHz
6LowPAN (802.15.4)	20–250 kb/s	10–20 m	2.4 GHz, 900 MHz, and 868 MHz
LoRaWAN	1 kb/s	10 km	900 MHz
4G LTE	100 Mb/s	30 km	800 MHz and 1900 MHz
5G	10 Gb/s	500 m	24 GHz–100 GHz

from/to any of the slave devices. One interesting note is that in this configuration, the slave device can only interact with the master and not with other slave devices. There are three main steps in Bluetooth's connection process:

1. Discovery: The Bluetooth devices run inquiries to discover each other and establish a relationship by exchanging specific information like the device name, and Bluetooth card address.
2. Pairing: After both devices have been discovered and have exchanged information, the pairing process starts. This process ensures the exchange of Bluetooth profiles between devices, which is stored in memory permanently. They also share a common secret key, which allows them to pair again in the future. Usually, pairing involves an authentication process that requires user intervention.
3. Connection: Finally, after devices are paired, they establish a permanent connection that permits the exchange of data. In connection mode, devices can be actively transmitting or receiving data, but also can be put in different sleep modes to save power.

- *Bluetooth Low Energy (BLE)*: Different from NFC and traditional Bluetooth, BLE is a communication protocol specifically tailored for IoT [79]. BLE implements almost the same communication capabilities (communication range and band) of traditional Bluetooth but with substantially reduced power consumption and cost. These differences benefit remote IoT applications that have limitations in power availability. Despite the communication similarities between Bluetooth and BLE, the later is not backward-compatible with Bluetooth, so devices that implement each protocol cannot form a common network. However, since they both share the same communication band, devices can implement BLE or Bluetooth using a common antenna which is convenient for IoT due to the expected space restrictions in resource-limited IoT devices.
- *WiFi*: This is a protocol suite based on the family of wireless standards IEEE 802.11. WiFi is broadly deployed and constitutes the *de facto* wireless technology for Internet communications. As early IoT integrated traditional communication protocols, WiFi is also widely deployed among IoT devices like smart doorbells, smart TVs, cars, and drones. In WiFi, compatible devices require a centralized entity (i.e., the router or hotspot) to network among them and the Internet. WiFi connections follow an "always-on" approach where channels can be shared between devices, but only one transmitter can send data on a specific channel at any moment in time. Because it is extensively used, WiFi is more vulnerable to cyber-attacks. Thus, authentication mechanisms are typically implemented to authorize new devices and users in the network [80,81].
- *ZigBee*: There are three different entities in the ZigBee network: the coordinator, the router, and the end device [82–84]. First, the ZigBee coordinator acts as a master device that controls and monitors the data transmission process. Second, the ZigBee router acts as a bridge that relays data to/from the Zigbee end devices. Such an architecture allows for ZigBee to favor high interoperability among several devices and to be one of the most popular wireless communication protocols used in IoT (e.g., commodity IoT and industrial IoT). As shown in Table 6, ZigBee is supported by the IEEE 802.15.4 standards, and its main goal is to provide connectivity to "nearby" devices with low latency and power consumption.
- *Z-Wave*: Zensys proposed the proprietary Z-Wave alliance in 2005 [85,86]. This technology allows for the low-power wireless connectivity among IoT devices, usually forming a mesh network. As Z-Wave is intended to permit high interoperability among different devices, specific protocols at the application layer allow different Z-Wave software and hardware to work together. Two of the main advantages offered by Z-Wave communications relays on its proposed architecture which permit for enhanced coverage and reliability. As Z-Wave devices use intermediate nodes to avoid obstacles between communication links, the network minimizes the "dark" communication spots that may appear in smart home setups. Thus, even though the simplest setup only includes a pair controllable-controller, additional devices can be added at any time following a similar approach as in Bluetooth. That is, a new device must be "added" first into the network (i.e., paired) before

it can communicate via Z-Wave. Also, as in Bluetooth, the pairing or addition process must be done only once.

- *6LowPAN*: This protocol implements IPv6 technology on Low-Power Wireless Personal Area Networks (LowPAN) and has been proposed specifically to be used in IoT by the Internet Engineering Task Force (IETF) [87,88]. The motivation behind 6LowPAN is to provide IP IoT-necessary connectivity to devices with minimal computing and power capabilities. As ZigBee, this protocol is implemented over 802.15.4 standards and provides low transmission throughput in short-range communications.
- *LoRa and LoRaWAN*: These protocols enable long-range communications with very-low power consumption. On the one hand, LoRa enables the long-range communication link. On the other hand, LoRaWAN is mainly responsible for managing the communication frequencies, data rate, and power for all the IoT devices [89,90]. Specifically, LoRaWAN is designed to define the upper layers of networked IoT devices. As a cloud-based Medium Access Control (MAC) layer protocol, LoRaWAN is mainly utilized to support and manage the communication between IoT gateways and end devices. LoRa-based communications are generally asynchronous, so end devices transmit their data to multiple IoT gateways whenever the data is available. Then, the gateways forward the data to a centralized network server.
- *Long-term Evolution (LTE) and 5G*: several IoT researchers and manufactures envision LTE and/or 5G as the future for IoT communications. In fact, the integration of new IoT-enabled LTE/5G communication chipsets into modern IoT devices and the deployment of new LTE/5G platforms for IoT [91,92] (i.e., LTE IoT) are driving the expansion of modern cellular networks over the IoT ecosystem. In general, LTE/5G implementation in IoT allows for more flexible, efficient, high throughput, long-range, and low-cost solutions, which are very attractive for manufacturers and IoT platforms. More details are presented in Table 6.

Advantages: The extended support to additional protocols gives the IoT user complementary tools to set up a more efficient and interoperative IoT solution. As with a higher number of IoT devices, it would be of benefit to the IoT user to set up their IoT environment with the required features so she can fulfill more complex applications. Given that any IoT device could use any number of protocols, it would make sense for an IoT platform to introduce means for which new protocols can be supported within the platform. *Disadvantages:* Providing support for protocol development is perhaps the most difficult and dangerous of third-party support options. To be widely used, a protocol must go through rigorous standardization processes, which include bug testing and security evaluations. From there, it can be adopted by IoT platforms directly where all protocol specifications are met. Allowing a user to implement her own protocol from scratch on her IoT solution could cause communication bugs or security faults that would impact her entire IoT solution. Also, implementing protocols is not a straightforward task and would require a deep technical background from the user.

5.5.1. Comparative analysis and summary

In Table 7, we show how the different IoT platforms offer extended protocol support to the users for a better experience while implementing their IoT solutions. For each platform, a full circle entails that a user can utilize available APIs or SDKs offered by the specific platforms to provide extended protocol support within the IoT solution. One would expect sandboxed and open-source platforms to be more reluctant to offer extended protocol support as sandboxed platforms always limit the set of available APIs, and open-source tends to pass the development burden to the final user. However, even though openHAB is an open-source platform, it offers a dedicated set of APIs for extended protocol support to the users, something that is not available in other (even) proprietary platforms like Apple HomeKit. Another interesting finding is that, contrarily to the design logic previously exposed, sandboxed platforms like ThingWorx are also able to facilitate the development process in implementing customized protocols for the user.

5.6. Security of the IoT Solutions

Security vulnerabilities may be present and cyber-attacks may occur at any layer of an IoT solution [93–95]. IoT devices may have embedded hardware trojans, communications can be disrupted/blocked to implement a denial of service attack, or data can be altered/stolen/corrupted, to name a few examples. In this section, rather than providing advantages and disadvantages of an IoT platform regarding security, we evaluate security issues and vulnerabilities that may affect the IoT solution. When applicable, we address if the specific threat or vulnerability has a known (or knowable) fix or solution that could be potentially implemented by some of the IoT platforms analyzed. We evaluate which IoT platforms are doing better in implementing current security mechanisms to protect the integrity of the IoT solution, and highlight the steps that individual platforms take "above and beyond" to guarantee and improve IoT security. We mainly study security vulnerabilities and cyber-attacks affecting different layers of the IoT solution that impact the principles of the essential security services of confidentiality, integrity, availability, and access control. Finally, we highlight current research works that can be applied to improve IoT platform security.

Confidentiality. Sensitive IoT data (e.g., health/financial information, device states, user behavior-related data) [7] and devices need to be protected from malicious actors. However, the nature of IoT (i.e., low computational resources, diversity of services, cloud-based implementation) leads to vulnerabilities affecting confidentiality. For instance, at the lowest layer of the IoT solutions, devices and sensors are vulnerable to side-channel, malicious code injection, and eavesdropping types of attacks. Specifically, in the case of a side-channel attack, a malicious actor may gain unauthorized access to the IoT system and infer sensitive information only by studying the changes in power consumption and electromagnetic emanation from the devices' hardware. Also, IoT devices are vulnerable to malicious code injection into their firmware, which can be used to change the devices' behavior to the attacker's advantage and facilitate elevated unauthorized access. Finally, IoT devices and sensors may leak sensitive information while performing unsecure authentication and key exchange mechanisms that are vulnerable to brute-force attacks [96–98]. Furthermore, the network layer is also vulnerable to attacks that affect the confidentiality of the IoT solution. For instance, phishing campaigns targeting IoT users may open the door to Advanced Persistent Threats (ATP) affecting the IoT networks that facilitate other attacks like sensitive information leakage and behavioral fingerprinting of users and devices [99,100]. The data processing layer is also vulnerable to attacks affecting the confidentiality of the IoT solution. Specifically, cloud services can be targeted by Man-in-the-Middle (MiM) and Database Injection Attacks. In the first attack, an unauthorized entity may be able to exploit network protocol vulnerabilities (e.g., MQTT) to intercept cloud communication steal IoT credentials [101]. Also, an attacker can bypass authentication mechanisms used in vulnerable database schemes to inject special queries and steal authentication data from users, devices, and apps. Finally, the IoT applications layer is also vulnerable to attacks affecting confidentiality. More precisely, malicious actors may use sniffer applications and devices to steal confidential information being sent during authentication services over the IoT network [102].

Manufacturers of IoT devices, especially those being utilized within critical infrastructure, are encouraged to follow best security practices to protect from side-channel attacks. Some of these practices include providing proper shielding against heat and electromagnetic emissions. Also, IoT platforms take two specific steps to protect from cyber attacks affecting the confidentiality: (1) the use of encryption and (2) proper authentication mechanisms. However, we note that these efforts are largely challenged by the lightweight design of IoT devices and the mass proliferation of IoT devices in networks.

Table 7
Capability of the different surveyed IoT platforms of providing extended protocol support. Platforms that facilitate the implementation of a higher number of protocols may provide extended usability to the user (●= Required, ◐= Optional, ○= Not Used).

IoT Platform extended protocol support	
Platform	Extended protocol support
openHAB	●
SmartThings	○
HomeKit	○
Windows IoT	●
FarmBeats	○
AWS IoT	○
ThingWorx	●
Watson IoT	○

- *Encryption Mechanisms*: The lightweight nature of IoT devices leads to unbearable performance costs when implementing encryption on top of the IoT communications [103] as compared to modern computers. This leads to a natural dichotomy between the IoT device's performance and the necessary confidentiality of its data. To maintain the IoT devices' expected performance, weaker encryption standards may be utilized in the communication between devices [104]. As IoT communication links are protected with weaker encryption standards, attackers have increased opportunities to reveal the communication content via the discovery of the encryption keys. For example, it has been noted that through bug exploitation, attacks on Z-wave-based devices from the SmartThings platform can downgrade the Z-wave protocol to reveal the encryption key [105], allowing for attackers to reveal any data being exchanged to/from those devices. IoT platforms are currently enforcing stronger encryption mechanisms in critical areas of the IoT solution (e.g., hubs, data storage units, administrative control units, etc.). Such improvements in security would only propagate to the end devices as they gain computation power. In the comparative analysis (Section 5.6.3), we highlight the *currently enforced* encryption standards in each IoT platform.
- *Authentication Mechanisms*: There are many dozens of authentication methods that can be used to validate entities accessing the IoT solution [14,106–108]. Many current IoT platforms adopt token-based authentication as a means to verify authorized IoT devices. While far from perfect due to scalability issues and additional overhead, tokens are a step in the right direction as multiple forms of authentication constitute a necessity in modern networks, IoT included. Vast amounts of work have been done to develop novel authentication mechanisms for IoT. However, further work must be done to develop policy, metrics, and adaptability of these methods. In our comparative analysis (Section 5.6.3), we discuss different forms of authentication mechanisms adopted by the surveyed IoT platforms so the user of the IoT solution can select the one that works the best for every specific application.

Integrity. IoT devices and the data they handle must be tamper-proof as not to produce falsified information. We find that there are specific issues with ensuring integrity in IoT that can be highlighted in this manuscript. These issues are directly related to the scale (in terms of the number of devices and data amounts) and the complexity of the modern IoT solutions. Also, these issues mark the current and future research and development works regarding IoT integrity. In the following, we overview these integrity issues in IoT.

- *IoT Devices Layer*: The lack of trusted manufactures and the mass proliferation of IoT devices naturally leads to the issue of hardware being developed by untrustworthy authorities. Given that small chips can be easily embedded into manufactured devices [109], the possibility of *rogue hardware* in an IoT solution increases drastically. Rogue hardware embedded into an IoT solution would effectively have the capacity to corrupt IoT data as it sees fit and insert false measurement into the network (i.e., false data injection attacks). Research into detection, isolation, and recovery from malicious hardware [13,14,73,110–115] would be of massive benefit to IoT. Node capturing attacks also affect the integrity of IoT devices. In these attacks, the malicious actors may try to remove legitimate devices from the network and replace them with fake/unauthorized devices that may look legitimate but, are fully controlled by the attacker [96]. Future work in hardware security for IoT devices would prove useful for this reason. One direction that research could go is finding a way to help automate the process of detecting compromised devices either in the supply chain and in the field. Another related research to hardware security in IoT would be developing anti-tampering methods for hardware manufacturing. Knowing that a device was not tampered during manufacturing or assembly time would go a long way toward ensuring the hardware security of individual IoT devices. Finally, effective protection against IoT rogue hardware would not be possible without full cooperation among vendors and researchers. IoT platforms that encourage and facilitate such cooperation would be a step ahead in terms of hardware security in IoT.
- *Communication Protocols Layer*: As mentioned, the publisher/subscriber is the most widely adopted method for event handling in IoT, with the MQTT protocol its most successful implementation. Pasknel notes that unsecured MQTT brokers are prone to man-in-the-middle attacks, allowing attackers to falsify IoT data flowing through the network [116]. To counter this, MQTT implements TLS support to verify parties in an IoT transaction [48]. However, as of this writing, TLS is not enabled (or enforced) by default, requiring users to add TLS support in their IoT solutions manually. If the user is not aware of this requirement or is not technically savvy, the IoT solution will be vulnerable to man-in-the-middle type of attacks.
- *Data Processing Layer*: Cloud malware injection attacks mainly cause attacks affecting the integrity of the data and the IoT solution in general within this layer. In this case, the attacker may inject malicious code or even an unauthorized virtual machine into the cloud service, pretending to be a legitimate service. With this, the attacker may be able to access sensitive data and further change it or destroy it. There are several steps IoT platforms can follow to prevent cloud malware injection attacks affecting their remote services. The most popular countermeasure is the use of secure regions or Hypervisors that regularly check the integrity of the entire cloud system.
- *IoT Application Layer*: Different programming languages bring their security vulnerabilities to the IoT solution. For example, Samsung SmartThings allows the use of Groovy-specific features in the IoT development like dynamic method invocation and state (global) variables. Recent research works have shown how these Groovy-specific challenges can impact the security and privacy of IoT apps, making the analysis of IoT applications difficult [9]. Carelessly developed apps may introduce well-known

security vulnerabilities from all the different programming languages into the IoT solution. For instance, some programming languages like C are intrinsically vulnerable to software security flaws that create opportunities for attackers to leak information or insert compromised code that changes the behavior of the IoT apps [117]. Several previous research works have demonstrated how the inclusion of an unsafe function in C can be utilized by malicious entities to perform attacks such as buffer overflow and other memory-based attacks. On the other hand, programming languages like Java, while relatively more secure than C, have other different vulnerabilities or are commonly involved in well-known malpractices (e.g., the overuse of public classes, method, and variables; the use of constructors to initialize variables). These Java-based programming malpractices may lead to security problems like code or command injection [118,119]. An IoT platform allowing third-party software makes it easier for an attacker to infiltrate an IoT solution with malicious software. All it would take is a user to install a malicious application on her IoT solution for an attacker to successfully infiltrate the solution. Some platforms, such as HomeKit, mitigate this issue by vetting third-party applications before they are allowed on their app store. However, this vetting process is not always guaranteed. For an IoT platform to improve software and application security, the most straightforward approach would be to limit the third-party support of IoT applications. Nonetheless, as we discussed earlier in this survey, such an approach would also mostly limit the functionality or applicability of the IoT solution. Research works have been done to improve application and software security in IoT. Instead of redesigning programming languages from the ground up, developers and researchers are proposing tools to statically [9,120] or dynamically [7,121,122] analyze IoT applications before they are utilized in an IoT solution. IoT platforms facilitating the development and integration of similar analysis tools would be one step ahead in the race of guaranteeing the expected software security to the IoT solution and of gaining more users.

5.6.1. Availability

Users of an IoT solution desire the benefit of "always-on" services. Specifically, in critical infrastructures such as Healthcare IoT or Industrial IoT, the data from the IoT solution must be available 24/7 to the users [123]. Availability in IoT must be considered and implemented in all hardware, software, and network, and data levels. IoT programming platforms that implement "always-on" services (e.g., AWS IoT, Homekit, and Watson IoT) get the immediate benefit of high-availability access to the IoT data. Users of these IoT programming platforms can upload their data to the provided cloud service and trust that their data will be readily accessible compared to hub-based systems where users must ensure this availability themselves. On the other hand, a significant issue with all IoT solutions relates to the availability of the IoT devices. Current industrial machines are expected to work for decades in their current environment. This concept would, therefore, apply to IoT-enabled version of the traditional industrial settings. Since IoT devices are often lightweight by design, it quickly becomes challenging to ensure that all IoT devices will be available when needed. Several threats can disrupt the availability of the different components of the IoT solution. For instance, at the hardware level, attackers can induce sleep deprivation attacks to speed-drain the battery of low-powered IoT devices [93]. At large, these types of attacks would make the IoT device completely unavailable, a critical outcome, especially for the remotely-located devices. Denial of Services attacks (DoS) and Routing attacks are the most widespread threats to the availability of the IoT services at the network level. In these cases, attackers flood the target servers with unwanted network requests and or redirect the path of the data flowing through the network, respectively [94]. Similarly, flooding attacks can be implemented to disrupt the availability of IoT data processing layers. Finally, disrupting devices, networks, and cloud servers directly impact the availability of IoT applications. For instance, if a remote service is unavailable, an IoT application would not be able to share/receive specific data and commands necessary for executing its functionality.

To mitigate the risk of service unavailability, IoT platforms are following architectural paradigms that favor configurations with redundant configurations of IoT devices being used in critical roles to add fail-over and fault tolerance capabilities — something many companies may not implement fully or correctly. Also, tampered or compromised devices constitute a correlated issue to a device's failure as frequent attacks look into causing a denial of services via device failure or malfunction [7,13,14]. As mentioned before, IoT researchers are proposing methods to detect when devices are tampered or failing before faults occur (e.g., tamper-resistant algorithms and classification), and implementing them in working IoT solutions is heavily desired.

Access Control. Access control in IoT solutions follows the same principles as access control in any other system — it provides a way to limit or control the capabilities of the entities accessing the solution. Specifically, we want to have the ability to monitor and control how users, devices, and applications use and perform tasks within the IoT solution. We discuss methods and best-practices for granting access to an IoT solution.

For the IoT platforms that we are considering in this survey, access control is implemented similarly. While the exact term used to define control is different in every platform, entities inside the IoT solution have defined specific *capabilities* that define the way they function and the type of task they can perform (e.g., a lightbulb having ON/OFF, dimming, and color controls). Then, the first goal of access control is to grant the right entity access to their specific capabilities. A significant security flaw in IoT's access control is the simplicity of adding new entities to the solution and granting access to those entities to the different users. SmartThings is particularly vulnerable to this problem, where it has been observed that it is possible for rogue users (e.g., an Airbnb guest) to add new entities (i.e., devices) to the IoT solution (i.e., smart home environment where they are rented) [16,118]. Necessarily, an external user can be granted permission to use the SmartThings application, which also carries the right to adding new devices and users to the solution as she sees fit. Further, the new "added" users would have the same access rights as the homeowner, including the means to define new policies on the use of the devices that would potentially conflict with the ones defined by the owner [16]. This access control problem, known as *over-privileged users and devices* is widespread across all of IoT, and IoT users must consider it before designing their solutions. Another major issue with access control in IoT is the complete lack of granularity in IoT solutions. When IoT platforms properly implement role-based access control with fine-grained permissions, the individual IoT solution becomes more secure. However, we find that in the current state of IoT, most IoT platforms have coarse-grained access control definitions (i.e., a user has full access to all devices in a smart-home environment), leading to security problems such as over-privileged and role-escalation.

Thus, the most robust way of implementing access control in IoT that we have observed occurs when platforms define specific capabilities for every entity in the IoT solution. Amazon AWS IoT is an example of an IoT platform that follows this approach [33]. When a new entity is created in an AWS IoT implementation, the IoT administrator grants the entity capabilities on a case-by-case basis. For instance, a new user in an AWS IoT solution may only be granted to view the IoT data related to specific devices. This allows for a more fine-grained methodology to grant access to the IoT solution as it allows administrators to develop specific access control policies for the entire IoT solution.

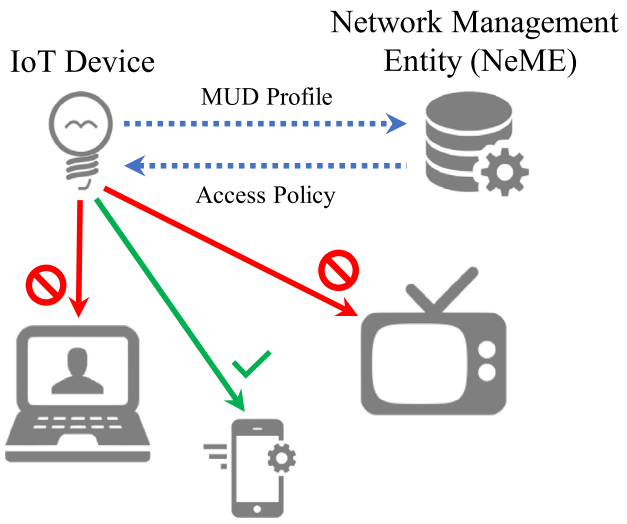


Fig. 4. The diagram demonstrates a high-level implementation of MUD specification for IoT. Based on its functionality, the smart bulb is allowed to communicate with the smartphone that has the bulb's controlling app installed. However, the network manager stop the bulb from communicating with other devices.

5.6.2. Manufacturer Usage Description (MUD) specification

In an effort to provide access control and guarantee proper connectivity within heterogeneous IoT solutions, the Internet Engineering Task Force (IETF) has proposed the Manufacturer Usage Description (MUD) specification [124]. The primary purpose of MUD is to "inform" the Network Manager Entity (NeME) about specific needs in terms of access and network functionality the IoT devices would require to function according to the specific application they support within the IoT solution [125].

Fig. 4 depicts a high-level implementation of the MUD specification within an IoT solution. In this case, a smart bulb (i.e., edge IoT device) shares its MUD profile to the NeME. In return, the NeME sends back the network access policy that governs the bulb's behavior within the solution, based on its specific application. For instance, the bulb is allowed to communicate with a smartphone with the bulb's controlling app installed. However, the same network administration entity prevents the bulb from communicating with other devices (i.e., Tv, laptop connected via SMTP port) that performs functionality outside of the bulb's behavior. In general, every type of IoT device would be required to have a MUD signature. Once the device is connected to the network, the MUD signature is sent to the NeME via a network request of type Link Layer Discover Protocol (LLDP), Dynamic Host Configuration Protocol (DHCP), or 802.1X (depending on the types of protocols the device supports). Then, the NeME extracts the MUD specification and forwards it to the MUD controller, which is in charge of extracting the context-specific policy information of the IoT device. Such a policy is then enforced within the network in the form of port-specific Access Control Lists (ACLs) [126–128].

Beyond access control, IoT platforms may take advantage of the MUD specification benefits to protect their IoT solutions from cyber attackers. First, by implementing MUD, the platforms directly reduce the impact of cyberattacks. Specifically, compromised IoT devices would have access to only specific network ports, which reduce the ways an external attacker can exploit a known vulnerability. Second, network administrators may use the information provided by the MUD to implement specific Intrusion Detection Systems (IDS) based on, for instance, analyzing the behavior of the devices while they try to violate the policies specified by MUD. Finally, IoT platforms may also benefit from MUD as these policies further facilitate the integration of IoT devices from different manufacturers to heterogeneous solutions. Recent research works have also demonstrated how MUD can be utilized to create a network abstraction layer among the IoT devices from different manufactures and platforms. Such an abstraction would allow for enhanced interoperability of the IoT solution while improving security [129–131]. In this context, Cisco is pioneering the manufacturing of MUD-capable network controller to protect IoT solutions at network-level [128,132,133]. IoT platforms may follow Cisco guidelines into incorporating MUD support and capabilities; however, some technical challenges need to be overcome. For instance, the Dynamic Host Protocol Configuration (DHCP) client in IoT devices needs to be updated, so it supports MUD. Also, configuring a Cisco MUD controller may be difficult for the user and require the acquisition of rather expensive equipment [134]. There exist some open-source MUD managers like osMUD [135] that could be more easily incorporated by the IoT platforms, however, these developments still in very early stages.

5.6.3. Comparative analysis and summary

As a baseline, IoT platforms should incorporate standard security practices that are easy to incorporate into the IoT solution by the users (e.g., fine-grained access control, multiple authentication methods, strong encryption). Such practices would prevent common attacks from occurring in the IoT solutions. In Table 8, we compare these standard practices against the surveyed IoT platforms. For access control, we define fine-grained access control as the ability to define and control any capability (e.g., read, write, execute on IoT data) in any entity (device or user). We note that most IoT platforms do not currently have this approach fully implemented. For authentication methods, we note any methodology introduced by the IoT platform that goes beyond standard encrypted communication (e.g., openHAB introduces an individualized SSL certificate authority unique to the IoT solution). Finally, for cryptography, we detail the recommended level of cryptography for each platform. One can observe that all platforms implement SSL/TLSv1.2 to secure communications; however, every platform implements different levels of cipher suites, which may impact the specific application of the IoT solution.

5.7. Privacy

IoT devices collect massive amounts of user information for data analysis purposes [8,12]. The data collected is used to improve the performance of the platform and make devices operate more efficiently while offering users tailored services depending on their behavior (e.g., users may receive advertisement about nearby restaurants right around the time they normally have dinner at home every day). However, some data may be too personal, making people hesitant from sharing it. In this section, we survey at the type of user data that IoT platforms collect, how the data is shared (e.g., with third-party services), and how this process may impact a user's privacy.

Privacy Issues within the IoT Solution. The IoT solutions handle and have access to a diverse set of sensitive information that, if leaked, may compromise the privacy of the users. Privacy issues may occur at any layer of the IoT solution (Fig. 1). For instance, compromised devices may use back-doors or malicious pieces of kernel code to send sensitive data to external unauthorized parties [13,14]. Also, savvy attackers may passively observe IoT communications (even when they are encrypted) to infer users activities and fingerprint their behavior [23]. Further, privacy concerns can also be found in the data processing layer, as service providers may share information related to the IoT solutions with third-parties without informing the user [12]. Finally, the application layer can also represent privacy concerns to the IoT user. Several research works have demonstrated that market or malicious IoT apps may compromised the IoT user's privacy in several ways. For instance, IoT applications can share sensitive data with external servers without informing the user, or malicious apps can use code to leak sensitive data to attackers [8,9,136]. As discussed in

Table 8
We compare IoT platforms regarding the mechanisms they offer to implement security via protecting the confidentiality, integrity, and access control (●= Required, ◐= Optional, ○= Not Used).

IoT Platform security methods comparison			
Platform	Fine-grained access control	Authentication methods	Recommended cipher suite
openHAB	●	Unique SSL certificate authority	256-bit ECC, ECDHE, ECDSA, AES256, GCM, SHA-384
SmartThings	○	REST access tokens	ECDHE, RSA, 3DES, EDE, CBC, SHA256
HomeKit	○	End-to-end encryption	RSA2048, AES256, ECC256, SHA256
Windows IoT Core	○	SecureBoot, Trusted Platform Modules	ECDHE, RSA, AES256, CBC, SHA384
FarmBeats	○	SecureBoot, Trusted Platform Modules	ECDHE, RSA, AES256, CBC, SHA384
AWS IoT	●	Individual certificates, REST access tokens	SHA-256, ECDHE, ECDSA, AES128, GCM
ThingWorx	○	Directory service authentication	RSA, AES256, CBC, SHA256
Watson IoT	●	Individual certificates, REST access tokens	ECDHE, RSA, AES128, GCM, SHA-256

Section 5.1, some IoT platforms implement cloud-based topologies to permit off-site data storage and processing. In these cases, user privacy becomes a particular issue. Besides the company hosting the cloud service potentially selling the user data for advertising purposes [12], there is also the added threat of the cloud service being breached, leaking user data to malicious parties. Generally, if an IoT platform stores IoT data on its cloud service, the user is expected to trust the IoT platform to ensure her data is kept private. However, most platforms do not inform the user regarding the type of information that is being collected from the IoT solution and who this information is being shared to. Thus, the IoT user does not have the choice to make informed decisions regarding the use of IoT devices, applications, and platforms [8]. To gain user trust, some IoT platforms are using new technologies that leave data stored on the cloud still under control of the user. Watson IoT implements Blockchain IoT [137] that allows a user to save her IoT data in a traditional blockchain-style ledger. After creating the ledger, the user chooses what sensitive data is to be stored in the blockchain. From there, the user can allow trusted parties to contribute to the blockchain. Then, IoT data from all parties are pooled into blocks, which are verified and stored permanently into the blockchain ledger. This blockchain-based concept allows the data to be stored on IBM cloud services while still keeping the privacy of the data in control of the user. Additionally, this concept provides a level of integrity to the IoT data — all parties in the blockchain must validate the data before it is stored into the blockchain.

Over-privileged Devices. Some devices may collect data and send it back to the hardware manufacturer. Under the platforms that allow soft-grained access control, these devices would have unfettered access to store user data by default [138] - requiring the user to acknowledge what type of data each device is capable of sending to third-parties so she can take action on it. With the platforms that follow a least-privilege approach to access control, the devices would have to be manually allowed to send and store user data — ensuring the user is aware of what type of data she allows to be handled by each device. The downside of this "more secure" approach is that, in most cases, users are not directly informed about the type of data that the IoT solution requires to implement its functionality, which may impact the general performance of the solution [8,9].

IoT Device Reconnaissance. The first stage of any cyber attack is to perform *reconnaissance* [139] to discover any weaknesses in an IoT solution. The numerous IoT devices introduced into an IoT solution naturally leads to more attack vectors in the network. An attacker may then choose to discover IoT devices in a network [140] to abuse any unique vulnerabilities. From there, the attacker may escalate their privileges in the network to continue their attack. Approaches like timely patching and vulnerability scanning may prevent attack escalation from occurring on individual devices. Additionally, closing unneeded ports on IoT devices in the network limits the total number of attack vectors. However, we find that little has been done as of this writing to limit the effects of device discovery as it limits the natural and expected proliferation of IoT devices in networks.

IoT Activity Inference. Similar to device discovery, it is possible for an attacker to view the activities that individual IoT devices are performing (even when communications are encrypted) [23], which can be used to fingerprint user behavior. For instance, in a smart-home environment, it is possible for an adversarial onlooker to determine what the owner of the house may be done by inferring device activities in the wireless network. Beyond the obvious privacy breach, the attacker can use the gained information to determine when the owner may be out of the house to cause further harm (e.g., breaking, robbing, installing malicious devices) to the smart home. To mitigate this potential privacy breach, IoT platforms could introduce noise and meaningless events into the solution so that it may confuse the data collection process done by the attacker. As evidenced by Acar et al. [23], introducing fabricated IoT traffic in the network drastically reduces the accuracy and precision to classify IoT activities properly.

Information Leakage. In addition to device/activity discovery, IoT is particularly prone to information leakage where a user's private data is revealed. This especially becomes an issue in settings like hospitals and financial institutions where sensitive data must be protected under law. There is no current best practice to ensure data privacy in an IoT setting, but there are multiple avenues of research in this topic, including IoT application analysis [8,9]. Homomorphic encryption [141], the addition of noise [142], and data aggregation [143] are some examples of established ways to preserve privacy on a traditional network [144,145]. One area of research would be to continue applying some of these privacy-preserving methods in a way that works for IoT solutions. Since IoT data should remain as unperturbed as possible for data analysis purposes, applying these privacy-preserving methods could pose interesting challenges. First, IoT platforms are expected to be against to report how the user data is collected and used publicly. Second, specific architectural characteristics in IoT (platforms prioritize high connectivity among devices over secure implementation) and the high diversity in IoT make traditional privacy mechanisms not suitable for IoT. IBM's blockchain approach is another interesting avenue of research for IoT privacy. While this method has been applied already, it is still a relatively new approach. Designing better, more efficient privacy solutions with blockchain would prove itself useful to IoT.

5.7.1. Comparative analysis and summary

We found that except for specific attempts (e.g., blockchain-based approach in IBM Watson IoT), most IoT platforms still lack effective measurements to protect user privacy. The actual nature of IoT, where so much information is available to favor the implementation of new and more effective applications, difficult the implementation of privacy mechanisms. In fact, such mechanisms would necessarily limit the performance and functionality of the IoT solutions. Another interesting finding is that probably because of its high market share, the majority

of the solutions that researchers are implementing to protect the security and privacy of the IoT solutions are being proposed for the Samsung SmartThings platform. We believe that IoT users must consider these findings while selecting the IoT platform for their solutions as they impact the amount of effort necessary to guarantee proper privacy preservation.

6. Lessons learned and final discussion

In this survey, we first provided criteria that permit the comparative analysis of IoT platforms from different application domains. This analysis would provide the IoT users a good starting point for a secure and effective implementation of their IoT solutions. Comparing IoT platforms from different application domains is not a straightforward task. As noted earlier, our main goal is to provide comparison criteria that apply to any platform regarding the specific domain. We summarize our findings and lessons learned with the following discussion.

Specific Application. We believe that the specific application for which the IoT solution is being implemented must be the first selection criteria for the user. As we have mentioned before, IoT solutions may have specific requirements depending on the application. For instance, IoT solutions used to monitor and control variables in agricultural farms may have specialized power consumption and connectivity characteristics suitable for remote environments. As a practical generalization of this survey, IoT users can utilize the comparison criteria here provided to first correlate platforms within the same application domain.

More Secure and Privacy-aware IoT. In general, IoT platforms permit the implementation of IoT solutions with adequate usability. That is, users may expect that platforms offer the necessary functionality for their specific applications. However, such a level of functionality often limits the amount of protection the same platforms can offer to the sensitive data and the privacy of users. We have shown how malicious actors can attack an IoT solution at the different levels of the IoT solution (starting from the use of unauthorized hardware components during the device manufacturing process to the injection of malicious code at the app level). Thus, selecting platforms that provide usability with an adequate level of security is key. Users must first focus on IoT platforms that offer ready-to-use security mechanisms suitable for the specific IoT solution. Then, they can learn additional methods to protect the final solution by taking advantage of other selection criteria, like third-party and extended protocol support.

Third-party and Extended Protocol Support. Third-party and extended protocol support allow for enhanced usability of the IoT solution. However, as discussed in this work, it also gives the IoT user additional tools to implement security- and privacy-enhancing mechanisms. Thus, we believe that the IoT user must look for these criteria before selecting the platform for her solution. For instance, users can use third-party hardware support to use devices that are trusted by the user, and that also offers similar functionality as the ones offered by the platform. We found that adding new devices or software functionality to an IoT solution is not simple, especially for those users that do not have a deep technical background. On the one hand, having IoT platforms that support third-party support of IoT devices, applications, and protocols permit the non-technical users to enhance security and privacy within the IoT solution by implementing more secure and practical functionality using plug-and-play APIs in a fraction of time. On the other hand, more technical users may use these capabilities to improve further the security tools offered by the platform. For instance, extended protocol support permits the application of more secure encryption mechanisms or stronger two-factor authentication.

Openness for enhanced security. Finally, IoT users must look for platforms that are "more friendly" to the application of additional external security mechanisms. Researchers have offered security methods for IoT that impact all layers of the IoT solution. However, these mechanisms tend to be offered and evaluated only for specific platforms. For instance, SaINT [9], IoTDots [7], and IoTWach [8] protect the security and privacy of a SmartThing solution. Recent works like iRuler [42] tend to provide security to implementations from multiple platforms. However, these solutions are limited to evaluate the impact of insecure IoT rules and only protect the upper layers of the IoT solutions.

7. Related work

IoT is an emerging technology that is applied in many diverse solutions. There is still work being done to classify and evaluate IoT platforms effectively. In this section, we systematically compare the related works with this survey following the methodology listed below:

- We selected previously published works that either (1) proposed generalized design and implementation models that apply to several IoT platforms or (2) directly compare different IoT platforms following a specific well-structured criteria.
- We used the features proposed in Section 4 to establish a generalized comparison criteria that we can use to compare the studied related work with our survey.

Table 9 shows that while most of the related work focus on how different IoT platforms handle security and privacy, other limited number of evaluation criteria (at most three items) were also covered in these works. However, our work is the most comprehensive survey to date that study and propose a practical framework for IoT users that covers seven different evaluation criteria including topology, programming languages, third-party support, extended protocol support, event handling, security, and privacy.

Mazhelis et al. evaluate IoT platforms with criteria focused on architecture, device integration, and platform implementation [146]. Through this, the authors provide an analysis of IoT platforms from the perspective of a potential IoT application developer. Riahi et al. provide a similar framework to evaluate IoT platforms, this time focused specifically on security [44]. Our paper takes concepts from the framework of Riahi and merges with the evaluation from Mazhelis to show how architecture choices from an IoT platform affect its security. Other works survey the IoT landscape. For instance, the work developed in [147,148] identifies general trends throughout the industry, discussing the protocols, devices, and platform architecture types that are being used in IoT platforms. Also, the authors in [149] survey specific platforms, focusing mostly on the architecture types and communication protocols used in the specific platforms. Finally, the work in [150] surveys the security and privacy of the Internet of Things. Specifically, it details the security and privacy challenges and limitations of different IoT architecture layers and technologies, focusing on the well-known security principles of data confidentiality, integrity, and availability. Although iit is a compelling analysis, the work does not offer a comprehensive comparison on how different IoT platforms handle security or are specifically vulnerable to these threats but focuses on providing a high-level overview of potential security vulnerabilities in IoT. Our survey, however, considers specific features that directly impact the design and implementation of IoT solutions. We perform the analysis on some of the most popular and widely used IoT platforms to highlight similarities and differences. Our survey strategy expands to a broader depth of the IoT platforms, allowing further discussion of the trends highlighted in the more general survey papers.

There has also been a large amount of research into the security of IoT platforms. As IoT platforms grow in size and number, security is paramount. However, it has fallen behind the growth of the industry considerably, prompting a large amount of research into the area. Some research works perform a survey of current security challenges facing IoT platforms. These papers provide a good analysis of security risks from the device level to the software level within IoT platforms [3,151,

Table 9
We compare related published works with this survey (●= Studied for Multiple IoT Platforms, ◐= Studied for a Single IoT Platform , ○= Not Studied).

Related work comparison analysis							
Related work	Topology	Prog. Lang.	Third-party Support	Extended protocol support	Event handling	Security	Privacy
Mazhelis et al. [146]	○	●	○	○	●	○	○
Riahi et al. [44]	○	○	○	○	○	●	●
Sethi et al. [147]	○	○	○	●	●	○	○
Vashi et al. [148]	○	○	○	○	○	●	○
Mazhelis et al. [149]	●	○	○	●	●	○	○
Mendez et al. [150]	●	○	○	●	●	●	●
Celik et al. [3]	●	○	○	○	●	●	●
Xu et al. [151]	○	○	○	○	○	●	○
Zhang et al. [152]	●	○	○	○	○	●	●
Ammar et al. [153]	●	●	○	●	○	●	●
This Work	●	●	●	●	●	●	●

154,155]. Discussing more in-depth, [152] provides a comprehensive study of a large number of specific security issues and exploits within IoT environments. This paper classifies and then analyzes a large number of reported security threats within IoT. The works in [45,151,156], discuss specific security exploit within IoT solutions. For instance, authors in [156] focus on the Samsung SmartThings platform to discuss multiple exploits of over-privileged that an adversary can use to obtain valuable information about the system. This again shows why it is important to understand the design and architecture of IoT platforms as adversaries can exploit the weaknesses in each of the solutions if IoT users are not careful. Finally, the work in [153] surveys the security of a sample group of IoT frameworks. Its analysis mainly focuses on detailing the specific proposed architecture for every platform, the tools offered for developing third-party smart apps, discussion on the compatible hardware, and their security features. Specifically, for the security analysis part, the work only considers the security services of authentication, access control, and ways to protect the communications within the IoT solution. Our research improves this work by incorporating communication- and privacy-related features, in addition to extended security, into the analysis. Also, we focus on the practical needs of the IoT users (i.e., IoT administrators, developers, and researchers) to guide our study.

Differences from Existing Works. Our work is different from the works above as we focus on real-world design choices in actively maintained, widely-used IoT platforms. We introduce and follow specific comparison criteria, considering the potential practical needs of active IoT users, developers, researchers, to evaluate and determine what currently established design choices would work well in the IoT world and what needs further development (particularly in communications, security, and privacy). Our analysis strategy helps to establish a baseline upon which the IoT research community can build to improve the security and privacy of real-world IoT platforms. Most importantly, our findings gives IoT users the design and implementation criteria to make informed decisions on the selection of specific IoT platforms to implement their solutions. We define a comprehensive evaluation framework that considers seven different technical comparison criteria: (1) topology design, (2) programming languages, (3) third-party support, (4) extended protocol support, (5) event handling, (6) security, and (7) privacy. From this perspective, the existing surveys and other related works do not provide such valuable insights to improve the implementation capabilities of IoT users, but solely focus on what is existing.

8. Conclusion

With the increasing popularity of IoT devices, several different companies are developing new platforms to manage and control the interaction between IoT and users. IoT apps are mainly used to capture and process the sensor data and to automate and execute tasks. In general, these IoT platforms provide the means for IoT devices to communicate with each other and to provide real-time analysis for users. Due to the wide variety of applications in IoT, these platforms vary in their target audience and uses; however, they all must still adhere to specific programming and architectural paradigms to be viable. In this work, we presented an in-depth analysis of the most popular IoT platforms from different application domains, including OpenHAB, Samsung SmartThings, Apple HomeKit, Windows IoT Core, Microsoft FarmBeats, Amazon AWS IoT, ThingWorx, and Watson IoT Platform. We mainly focused on evaluation criteria that highlight topics in their topology, programming languages, event handling, third-party support, security, and privacy. As per our knowledge, by the time of writing this manuscript, there was not any previous work that covered these features for the most popular IoT platforms. Overall, we find that there is still plenty of work needed to be done in IoT development. As a baseline, fine-grained access control and authentication for all entities (user, devices) with individual permissions and identities at every layer of an IoT solution should be implemented in today's IoT platform. We note that further research in IoT security and privacy is still highly desired in topics such as tamper-resistant hardware, IoT application analysis, IoT device/activity discovery, preventing information leakage, and general vulnerability discovery.

CRediT authorship contribution statement

Leonardo Babun: Conceptualization, Methodology, Validation, Investigation, Writing - original draft, Visualization, Supervision. **Kyle Denney:** Methodology, Writing - original draft, Visualization. **Z. Berkay Celik:** Methodology, Writing - review & editing. **Patrick McDaniel:** Writing - review & editing, Funding acquisition. **A. Selcuk Uluagac:** Project administration, Writing - review & editing, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors wish to thank Andrew Ruf and Dr. Hidayet Aksu for their comments in the earlier versions of this work. This work was partially supported by the U.S. National Science Foundation (Awards: NSF-CAREER-CNS-1453647, NSF-1663051, NSF-CNS-1718116, CNS-1564105, and CNS-1900873), and Cyber Florida Capacity Building Program, USA. The views expressed are those of the authors only.

References

[1] Mehavarunan, Digital marketer, the future of IOT: 4 predictions about the internet of things, 2019, https://thriveglobal.com/stories/the-future-of-iot-4-predictions-about-the-internet-of-things/. (Online; Accessed 9 March 2020).

[2] IoT developer trends 2017 edition, 2017, URL https://ianskerrett.wordpress.com/2017/04/19/IoT-developer-trends-2017-edition/. (Online; Accessed 9 March 2020).

[3] Z.B. Celik, E. Fernandes, E. Pauley, G. Tan, P. McDaniel, Program analysis of commodity IoT applications for security and privacy: Challenges and opportunities, ACM Comput. Surv. (2019).

[4] Z.B. Celik, P. McDaniel, G. Tan, Soteria: Automated IoT safety and security analysis, in: USENIX Annual Technical Conference, USENIX ATC, Boston, MA, 2018, 2018, URL https://www.usenix.org/system/files/conference/atc18/atc18-celik.pdf.

[5] Z.B. Celik, G. Tan, P. McDaniel, IoTGuard: Dynamic enforcement of security and safety policy in commodity IoT, in: Network and Distributed System Security Symposium, NDSS, San Diego, CA, 2019, 2019, February.

[6] A.K. Sikder, L. Babun, H. Aksu, A.S. Uluagac, Aegis: A context-aware security framework for smart home systems, in: Proceedings of the 35th Annual Computer Security Applications Conference, ACSAC 2019.

[7] L. Babun, A.K. Sikder, A. Acar, A.S. Uluagac, IoTdots: A digital forensics framework for smart environments, 2018, arXiv:arXiv:1809.00745. (Online; Accessed 9 March 2020).

[8] L. Babun, Z.B. Celik, P. McDaniel, A.S. Uluagac, Real-time analysis of privacy-(un)aware IOT applications, in: Proceedings on Privacy Enhancing Technologies, Vol. 2021, No. 1, Po/PETS, 2021.

[9] Z.B. Celik, L. Babun, A.K. Sikder, H. Aksu, G. Tan, P. McDaniel, A.S. Uluagac, Sensitive information tracking in commodity IoT, in: USENIX Security Symposium, Baltimore, MD, 2018, 2018, URL https://www.usenix.org/system/files/conference/usenixsecurity18/sec18-celik.pdf.

[10] A.K.M. Iqtidar Newaz, A.K. Sikder, L. Babun, A.S. Uluagac, HEKA: A novel intrusion detection system for attacks to personal medical devices, in: The 2020 IEEE Conference in Communications and Network Security.

[11] L. Babun, H. Aksu, L. Ryan, E. Bentley, K. Akkaya, A.S. Uluagac, Z-IoT: Passive device-class fingerprinting of Zigbee and Z-wave IoT devices, in: The IEEE International Conference on Communications, ICC, 2020.

[12] H. Aksu, L. Babun, M. Conti, G. Tolomei, A.S. Uluagac, Advertising in the IoT era: Vision and challenges, IEEE Commun. Mag. 56 (11) (2018) 138–144, http://dx.doi.org/10.1109/MCOM.2017.1700871.

[13] L. Babun, H. Aksu, A.S. Uluagac, Identifying counterfeit smart grid devices: A lightweight system level framework, in: 2017 IEEE International Conference on Communications, ICC, 2017, 2017, pp. 1–6, http://dx.doi.org/10.1109/ICC.2017.7996877.

[14] L. Babun, H. Aksu, A.S. Uluagac, A system-level behavioral detection framework for compromised CPS devices: Smart-grid, ACM Trans. Cyber-Phys. Syst. 2019 (2019) 1–28, http://doi.acm.org/10.1145/3355300.

[15] L. Atzori, A. Iera, G. Morabito, The internet of things: A survey, Comput. Netw. (2010) 2787–2805.

[16] A.K. Sikder, L. Babun, Z.B. Celik, A. Acar, H. Aksu, P. McDaniel, E. Kirda, A.S. Uluagac, KRATOS: Multi-user multi-device-aware access control system for the smart home, in: 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks, WiSec 2020.

[17] J. Granjal, E. Monteiro, J.S. Silva, Security for the internet of things: A survey of existing protocols and open research issues, IEEE Commun. Surv. Tutor. 17 (3) (2015) 1294–1312, http://dx.doi.org/10.1109/COMST.2015.2388550, 3rd Quarter.

[18] H. Choi, Brightics-IoT: key attractive features of enterprise targeted IoT platform, in: 2018 IEEE International Conference on Industrial Internet, ICII.

[19] N. Vignesh, R. Lohith, S.A. Kumar, H. Dagale, K. Sangeeta, An IOT based network management system for enterprise network, in: 2018 IEEE International Conference on Industrial Internet, ICII.

[20] CRESTON, Creston technical institute, 2020, https://www.crestron.com/training. (Online; Accessed 10 July 2020).

[21] Control4, Join our world-class automation brand today, 2020, https://www.control4.com/for/dealers/. (Online; Accessed 10 July 2020).

[22] Lutron, About the lighting control institute (LCI), 2020, https://www.lutron.com/en-US/Education-Training/Pages/LCI/AboutLCI/About.aspx. (Online; Accessed 10 July 2020).

[23] A. Acar, H. Fereidooni, T. Abera, A.K. Sikder, M. Miettinen, H. Aksu, M. Conti, A.-R. Sadeghi, A.S. Uluagac, Peek-a-boo: I see your smart home activities, even encrypted!, in: 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks, WiSec 2020.

[24] Amazon, AWS IoT partner solutions, 2017, https://aws.amazon.com/IoT/partner-solutions/. (Online; Accessed 9 March 2020).

[25] IoT platform comparison: How the 450 providers stack up, 2020, URL https://IoT-analytics.com/IoT-platform-comparison-how-providers-stack-up/. (Online; Accessed 9 March 2020).

[26] SmartThings supported IoT products (Devices), 2020, https://www.smartthings.com/products. (Online; Accessed 29 January 2020).

[27] SmartThings code review guidelines and best practices, 2020, http://docs.smartthings.com/en/latest/code-review-guidelines.html. (Online; Accessed 29 January 2020).

[28] OpenHAB Community, Openhab documentation, 2017, http://docs.openhab.org/index.html. (Online; Accessed 9 March 2020).

[29] Samsung, Smartthings developer documentation, 2017, http://docs.smartthings.com/en/latest/getting-started/overview.html. (Online; Accessed 9 March 2020).

[30] Apple, Apple homekit documentation, 2017, https://developer.apple.com/homekit/. (Online; Accessed 9 March 2020).

[31] Microsoft, Windows IoT core documentation, 2017, https://developer.microsoft.com/en-us/windows/IoT/explore/IoTcore. (Online; Accessed 9 March 2020).

[32] D. Vasisht, Z. Kapetanovic, J. Won, X. Jin, R. Chandra, S. Sinha, A. Kapoor, Farmbeats: An IoT platform for data-driven agriculture, 2017, URL https://www.microsoft.com/en-us/research/publication/farmbeats-IoT-platform-data-driven-agriculture/. (Online; Accessed 9 March 2020).

[33] Amazon, Amazon web services privacy, 2017, online, https://aws.amazon.com/privacy/. (Online; Accessed 9 March 2020).

[34] PTC, Thingworx developer portal, 2017, online, https://developer.thingworx.com/dashboard. (Online; Accessed 9 March 2020).

[35] IBM, Watson IoT platform, 2017, online, URL https://console.bluemix.net/docs/services/IoT/feature_overview.html. (Online; Accessed 9 March 2020).

[36] Microsoft, Universal windows platform documentation, 2017, https://docs.microsoft.com/en-us/windows/uwp/get-started/universal-application-platform-guide. (Online; Accessed 9 March 2020).

[37] Microsoft, Azure FarmBeats (preview), 2020, https://www.microsoft.com/en-in/campaign/azure-farmbeats/. (Online; Accessed 30 March 2020).

[38] IoTBench, 2020, URL https://github.com/IoTBench/. (Online; Accessed 9 March 2020).

[39] Leonardo Babun, Hidayet Aksu, Selcuk A. Uluagac, Detection of counterfeit and compromised devices using system and function call tracing techniques, 2018, (Miami, FL, US), https://www.osti.gov/biblio/1463864, July.

[40] Leonardo Babun, Hidayet Aksu, Selcuk A. Uluagac, Method of resource-limited device and device class identification using system and function call tracing techniques, performance, and statistical analysis, 2019, (Miami, FL, US), https://patents.google.com/patent/US10242193B1/en, March.

[41] Kyle Denney, Enes Erdin, Leonardo Babun, Selcuk A. Uluagac, Kemal Akkaya, Systems and methods for inhibiting threats to a computing environment, 2019, (Miami, FL, US), https://patents.google.com/patent/US20200356665A1/en, December 2020.

[42] Q. Wang, P. Datta, W. Yang, S. Liu, A. Bates, C.A. Gunter, Charting the attack surface of trigger-action iot platforms, in: Proceedings of 26th ACM Conference on Computer and Communications Security, 2019.

[43] What are android app permissions, and how do devs implement them? 2017, URL https://www.androidauthority.com/android-app-permissions-explained-642452/. (Online; Accessed 9 March 2020).

[44] A. Riahi, Y. Challal, E. Natalizio, Z. Chtourou, A. Bouabdallah, A systemic approach for IoT security, in: 2013 IEEE International Conference on Distributed Computing in Sensor Systems, Vol. 2013, 2013, pp. 351–355, http://dx.doi.org/10.1109/DCOSS.2013.78.

[45] E. Fernandes, J. Paupore, A. Rahmati, D. Simionato, M. Conti, A. Prakash, Flowfence: Practical data protection for emerging IoT application frameworks, in: 25th USENIX Security Symposium, Vol. 2016, USENIX Security 16, USENIX Association, Austin, TX, 2016, pp. 531–548, URL https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/fernandes.

[46] DSL overview, 2020, URL https://www.tldp.org/HOWTO/DSL-HOWTO/overview.html. (Online; Accessed 9 March 2020).

[47] MySQL documentation, URL https://dev.mysql.com/doc/.

[48] OASIS, MQTT version 3.1.1, 2017, http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html. (Online; Accessed 9 March 2020).

[49] AMPQ, 2020, URL https://www.amqp.org/. (Online; Accessed 9 March 2020).

[50] P. Bhimani, G. Panchal, Message delivery guarantee and status update of clients based on IoT-AMQP, Intell. Commun. Comput. Technol. (2018).

[51] XMPP, 2020, URL https://xmpp.org/. (Online; Accessed 9 March 2020).

[52] D. Conzon, T. Bolognesi, P. Brizzi, A. Lotito, R. Tomasi, M.A. Spirito, The virtus middleware: An XMPP-based architecture for secure IoT communications, in: 2012 21st International Conference on Computer Communications and Networks, ICCCN, 2012.

[53] M. Kirsche, R. Klauck, Unify to bridge gaps: bringing XMPP into the internet of things, in: 2012 IEEE International Conference on Pervasive Computing and Communications Workshops, 2012.

[54] DDS, 2020, URL http://portals.omg.org/dds/. (Online; Accessed 9 March 2020).

[55] P. Peniak, M. Franekova, Open communication protocols for integration of embedded systems within industry 4, in: 2015 International Conference on Applied Electronics, AE, Sep. 2015.

[56] A. Al-Fuqaha, A. Khreishah, M. Guizani, A. Rayes, M. Mohammadi, Toward better horizontal integration among IoT services, IEEE Commun. Mag. 53 (9) (2015) 72–79.

[57] STOMP, 2020, URL https://stomp.github.io/. (Online; Accessed 9 March 2020).

[58] ZeroQM, 2020, URL http://zeromq.org/. (Online; Accessed 9 March 2020).

[59] D. Happ, N. Karowski, T. Menzel, V. Handziski, A. Wolisz, Meeting IoT platform requirements with open pub/sub solutions, Ann. Telecommun. 72 (1) (2017) 41–52.

[60] CurveZMQ, CurveZMQ, 2020, URL http://curvezmq.org/. (Online; Accessed 9 March 2020).

[61] D.J. Bernstein, CurveCP: Usable security for the internet, 2020, URL http://curvecp.org/. (Online; Accessed 9 March 2020).

[62] D.J. Bernstein, T. Lange, P. Schwabe, NaCl: Networking and cryptography library, 2020, URL http://nacl.cr.yp.to/. (Online; Accessed 9 March 2020).

[63] Representational state transfer (REST), 2020, URL https://www.ics.uci.edu/fielding/pubs/dissertation/rest_arch_style.html. (Online; Accessed 9 March 2020).

[64] RF wireless world, 2020, URL http://www.rfwireless-world.com/Terminology/MQTT-vs-REST.html. (Online; Accessed 9 March 2020).

[65] H. Shi, N. Chen, R. Deters, Combining mobile and fog computing: using CoAP to link mobile device clouds with fog computing, in: 2015 IEEE International Conference on Data Science and Data Intensive Systems, 2015.

[66] M. Castro, A.J. Jara, A.F. Skarmeta, Enabling end-to- end CoAP-based communications for the web of things, J. Netw. Comput. Appl. 59 (2016) 230–236.

[67] S. Cirani, G. Ferrari, N. Iotti, M. Picone, The IoT hub: A fog node for seamless management of heterogeneous connected smart objects, in: 2015 12th Annual IEEE International Conference on Sensing, Communication, and Networking - Workshops, SECON Workshops, 2015.

[68] A. Capossele, V. Cervo, G. De Cicco, C. Petrioli, Security as a CoAP resource: an optimized DTLS implementation for the IoT, in: 2015 IEEE International Conference on Communications, ICC, 2015.

[69] WebSocket, 2020, URL https://www.websocket.org/. (Online; Accessed 9 March 2020).

[70] P. Kayal, H. Perros, A comparison of IoT application layer protocols through a smart parking implementation, in: 2017 20th Conference on Innovations in Clouds, Internet and Networks, ICIN, 2017.

[71] D. Mun, M.L. Dinh, Y. Kwon, An assessment of internet of things protocols for resource-constrained applications, in: 2016 IEEE 40th Annual Computer Software and Applications Conference, COMPSAC.

[72] N. Naik, Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP, in: 2017 IEEE International Systems Engineering Symposium, ISSE.

[73] F. Naseem, L. Babun, C. Kaygusuz, S.J. Moquin, C. Farnell, A. Mantooth, A.S. Uluagac, CSPoweR-Watch: A cyber-resilient residential power management system, in: 2019 International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData).

[74] IoT Standards & Protocols Guide 2018 comparisons on network, wireless comms, security, industrial, 2020, URL https://www.postscapes.com/internet-of-things-protocols/. (Online; Accessed 9 March 2020).

[75] G.M. Garner, M. Ouellette, M.J. Teener, Using an IEEE 802.1as network as a distributed IEEE 1588 boundary, ordinary, or transparent clock, in: 2010 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication, Vol. 2010, 2010, pp. 109–115, http://dx.doi.org/10.1109/ISPCS.2010.5609779.

[76] Near field communications (NFC) forum, 2020, URL https://nfc-forum.org/. (Online; Accessed 9 March 2020).

[77] Bluetooth communications, 2020, URL https://bluetooth.com/. (Online; Accessed 9 March 2020).

[78] S. Oriyano, WiFi and bluetooth security, IEEE courses, 2020, URL https://ieeexplore.ieee.org/courses/details/EDP457. (Online; Accessed 9 March 2020).

[79] F.J. Dian, A. Yousefi, S. Lim, A practical study on bluetooth low energy (BLE) throughput, in: 2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference, IEMCON.

[80] H. Peng, WIFI network information security analysis research, in: 2012 2nd International Conference on Consumer Electronics, Communications and Networks, CECNet.

[81] B. Potter, Wireless security's future, IEEE Secur. Priv. (2003).

[82] Zigbee Alliance, 2020, URL https://www.zigbee.org/zigbee-for-developers/zigbee-3-0/. (Online; Accessed 9 March 2020).

[83] P. Baronti, P. Pillai, V. Chook, S. Chessa, A. Gotta, Y. Hu, Wireless sensor networks: A survey on the state of the art and the 802.15.4 and ZigBee standards, Comput. Commun. (2007).

[84] Zigbee stack layers, 2020, URL https://www.digi.com/resources/documentation/Digidocs/90002002/~Content/Reference/r_zb_stack.htm?TocPath=zigbee%20networks%7C___3. (Online; Accessed 9 March 2020).

[85] Z-wave smart home products FAQ, 2020, URL https://www.z-wave.com/faq. (Online; Accessed 9 March 2020).

[86] Z-wave tutorial, 2020, URL https://iotpoint.wordpress.com/z-wave-tutorial/. (Online; Accessed 9 March 2020).

[87] X. Ma, W. Luo, The analysis of 6LowPAN technology, in: 2008 IEEE Pacific-Asia Workshop on Computational Intelligence and Industrial Application.

[88] C. Yum, Y. Beun, S. Kang, Y. Lee, J. Song, Methods to use 6LoWPAN in IPv4 network, in :The 9th International Conference on Advanced Communication Technology, 2007.

[89] A. Zourmand, A.L. Kun Hing, C. Wai Hung, M. AbdulRehman, Internet of things (IoT) using LoRa technology, in: 2019 IEEE International Conference on Automatic Control and Intelligent Systems, I2CACIS.

[90] A. Lavric, V. Popa, Internet of things and LoRa™ low-power wide-area networks: A survey, in: 2017 International Symposium on Signals, Circuits and Systems, ISSCS.

[91] M. Klymash, H. Beshley, M. Seliuchenko, T. Maksymyuk, Improving architecture of LTE mobile network for IoT services provisioning, in: 2017 2nd International Conference on Advanced Information and Communication Technologies, AICT.

[92] B. Finley, A. Vesselkov, Cellular IoT traffic characterization and evolution, in: 2019 IEEE 5th World Forum on Internet of Things, WF-IoT.

[93] F. Meneghello, M. Calore, D. Zucchetto, M. Polese, A. Zanella, IoT: Internet of threats? A survey of practical security vulnerabilities in real IoT devices, IEEE Internet Things J. (2019).

[94] M.A. Ferrag, L. Shu, A. Derhab, L. Maglaras, Security and privacy for green IoT-based agriculture: Review, blockchain solutions, and challenges, IEEE Access (2020).

[95] O. Arias, J. Wurm, K. Hoang, Y. Jin, Privacy and security in internet of things and wearable devices, IEEE Trans. Multi-Scale Comput. Syst. (2015).

[96] S. Kumar, S. Sahoo, A. Mahapatra, A.K. Swain, K.K. Mahapatra, Security enhancements to system on chip devices for IoT perception layer, in: Proc. IEEE Int. Symp. Nanoelectron. Inf. Syst., iNIS, 2017.

[97] C.H. Liao, H.H. Shuai, L.C. Wang, Eavesdropping prevention for heterogeneous internet of things systems, in: Proc. 15th IEEE Annu. Consum. Commun. Netw. Conf., CCNC, 2018.

[98] S. Salamatian, W. Huleihel, A. Beirami, A. Cohen, M. Medard, Why botnets work: Distributed brute-force attacks need no synchronization, IEEE Trans. Inf. Forensics Secur. (2019).

[99] APWG, Phishing activity trends report, 2019, https://docs.apwg.org/reports/apwg_trends_report_q4_2019.pdf. (Online; Accessed 10 July 2020).

[100] C. Li, C. Chen, A multi-stage control method application in the fight against phishing attacks, in: Proc. 26th Comput. Secur. Acad. Commun. Across Country, 2011.

[101] V. Hassija, V. Chamola, V. Saxena, D. Jain, P. Goyal, A survey on IoT security: Application areas, security threats, and solution architectures, IEEE Access (2019).

[102] S.N. Swamy, D. Jadhav, N. Kulkarni, A survey on IoT security: Application areas, security threats, and solution architectures, IEEE Access (2019).

[103] M. Abomhara, G.M. Køien, Cyber security and the internet of things: Vulnerabilities, threats, intruders and attacks, J. Cyber Secur. 4 (1) (2015) 65–88.

[104] S. Friedman, NIST lays groundwork for encrypting IoT devices, 2020, URL https://gcn.com/articles/2018/04/19/nist-IoT-security.aspx. (Online; Accessed 9 March 2020).

[105] T. Spring, Millions of IoT devices vulnerable to Z-wave downgrade attacks, researchers claim, 2018, URL https://threatpost.com/millions-of-IoT-devices-vulnerable-to-z-wave-downgrade-attacks-researchers-claim/132295/. (Online; Accessed 9 March 2020).

[106] K. Habib, A. Torjusen, W. Leister, A novel authentication framework based on bio-metric and radio fingerprinting for the IoT in eHealth, in: Proceedings of International Conference on Smart Systems, Devices and Technologies, Vol. 2014, SMART, 2014, pp. 32–37.

[107] A. Acar, H. Aksu, K. Akkaya, S.A. Uluagac, Method for Continuous User Authentication with Wearables, US Patent App. 15/674, 133, Sep. 11 2018, URL http://www.freepatentsonline.com/10075846.html.

[108] A. Acar, H. Aksu, A.S. Uluagac, K. Akkaya, Waca: Wearable-assisted continuous authentication, 2018, arXiv preprint arXiv:1802.10417.

[109] J. Roberston, M. Riley, The big hack: How China used a tiny chip to infiltrate U.S. companies, 2018, URL https://www.bloomberg.com/news/features/2018-10-04/the-big-hack-how-china-used-a-tiny-chip-to-infiltrate-america-s-top-companies. (Online; Accessed 9 March 2020).

[110] K. Denney, L. Babun, A.S. Uluagac, USB-watch: A generalized hardware-assisted insider threat detection framework, J. Hardw. Syst. Secur. (2020).

[111] K. Denney, E. Erdin, L. Babun, M. Vai, A.S. Uluagac, USB-watch: A dynamic hardware-assisted USB threat detection framework, in: Int. Conference on Security and Privacy in Communication Systems, SecureComm, 2019.

[112] J. Myers, L. Babun, E. Yao, S. Helble, P. Allen, MAD-IOT: Memory anomaly detection for the internet of things, in: The Workshop on Impact of Artificial Intelligence on Internet of Things, co-located with the IEEE Global Communications Conference, GLOBECOM, 2019.

[113] L. Puche, L. Babun, A.S. Uluagac, HDMI-Walk: attacking HDMI configuration networks via CEC one step at a time, in: The 2019 Annual Computer Security Applications Conference, ACSAC 35, 2019.

[114] L. Puche, L. Babun, A.S. Uluagac, HDMI-watch: Smart intrusion detection system against HDMI attacks, IEEE Trans. Netw. Sci. Eng. (2020).

[115] L. Puche, L. Babun, A. Ahmet, A. Akkaya, A.S. Uluagac, PoisonIvy: (In)secure practices of enterprise IoT systems in smart buildingss, in: BuildSys 2020.

[116] V. Pasknel, Hacking the IoT with MQTT – morphus labs, 2017, URL https://morphuslabs.com/hacking-the-IoT-with-mqtt-8edaf0d07b9b. (Online; Accessed 9 March 2020).

[117] Common vulnerabilities guide for C programmers, 2020, URL https://security.web.cern.ch/security/recommendations/en/codetools/c.shtml. (Online; Accessed 9 March 2020).
[118] E. Fernandes, A. Rahmati, K. Eykholt, A. Prakash, Internet of things security research: A Rehash of Old Ideas or New Intellectual Challenges? 2017, CoRR, abs/1705.08522, URL http://arxiv.org/abs/1705.08522.
[119] I. Soumenkov, Web based infection vector, 2013, URL https://securelist.com/miniduke-web-based-infection-vector/57622/. (Online; Accessed 9 March 2020).
[120] L. Babun, B. Celik, Saint analysis console, 2020, URL https://saint-project.appspot.com. (Online; Accessed 9 March 2020).
[121] L. Babun, Iotwatch console, 2020, URL https://iotwatch.appspot.com/. (Online; Accessed 9 March 2020).
[122] L. Babun, Iotdots instrumenter console, 2020, URL https://iotdots-modifier.appspot.com/. (Online; Accessed 9 March 2020).
[123] A.K.M. Iqtidar Newaz, Amit Kumar Sikder, Mohammad Ashiqur Rahman, A. Selcuk Uluagac, Healthguard: A machine learning-based security framework for smart healthcare systems, in: IEEE SNAMS 2019.
[124] A. Hamza, D. Ranathunga, H.H. Gharakheili, M. Roughan, V. Sivaraman, Clear as MUD: generating, validating and applying IoT behavioral profiles, in: Proceedings of the 2018 Workshop on IoT Security and Privacy.
[125] P. Watrobski, J. Klosterman, W. Barker, M. Souppaya, Methodology for characterizing network behavior of internet of things devices, NIST Cybersecurity White Paper, 2020.
[126] E. Lear, R. Droms, D. Romascanu, Manufacturer Usage Description Specification, IETF, Network Working Group, 2018, https://tools.ietf.org/id/draft-ietf-opsawg-mud-22.html#rfc.section.1.1. (Online; Accessed 10 July 2020).
[127] E. Lear, R. Droms, D. Romascanu, Manufacturer Usage Description Specification, IETF, Network Working Group, 2018, https://tools.ietf.org/html/rfc8520. (Online; Accessed 10 July 2020).
[128] DevNet cisco, what is MUD? 2020, https://developer.cisco.com/docs/mud/#what-is-mud/what-is-mud. (Online; Accessed 10 July 2020).
[129] S.N. Matheu, J.L. Hernández-Ramos, S. Pérez, A.F. Skarmeta, Extending MUD profiles through an automated IoT security testing methodology, IEEE Access (2019).
[130] A. Hamza, D. Ranathunga, H.H. Gharakheili, T. Benson, M. Roughan, V. Sivaraman, Verifying and monitoring IoTs network behavior using MUD profiles, 2020, arXiv:arXiv:1902.02484. (Online; Accessed 10 July 2020).
[131] A. Feraudo, P. Yadav, R. Mortier, T. Benson, P. Bellavista, J. Crowcroft, SoK: Beyond IoT MUD deployments – challenges and future directions, 2020, arXiv:arXiv:2004.08003. (Online; Accessed 10 July 2020).
[132] Cisco blogs - security, MUD is officially approved by IETF as an internet standard, and Cisco is launching MUD1.0 to protect your IoT devices, 2020, https://blogs.cisco.com/security/mud-is-officially-approved-by-ietf-as-an-internet-st{and}ard-and-cisco-is-launching-mud1-0-to-protect-your-iot-devices. (Online; Accessed 10 July 2020).
[133] Multichannel news, Cisco deal boosts MDU strategy, 2020, https://www.multichannel.com/news/cisco-deal-boosts-mdu-strategy-158535. (Online; Accessed 10 July 2020).
[134] M. Ayar, SANS Institute, DICE and MUD protocols for securing IoT devices, 2020, https://www.sans.org/reading-room/whitepapers/internet/paper/38980. (Online; Accessed 10 July 2020).
[135] osMUD, Open source MUD manager, 2020, https://osmud.org/. (Online; Accessed 10 July 2020).
[136] Z.B. Celik, P. McDaniel, G. Tan, L. Babun, A.S. Uluagac, Verifying internet of things safety and security in physical spaces, IEEE Secur. Priv. (2019).
[137] IBM, Blockchain IoT, 2017, online, https://www.ibm.com/internet-of-things/spotlight/blockchain (Online; Accessed 9 March 2020).
[138] Y.J. Jia, Q.A. Chen, S. Wang, A. Rahmati, E. Fernandes, Z.M. Mao, A. Prakash, S.J. Unviersity, ContexIoT: Towards providing contextual integrity to appified IoT platforms, 2017, 2017.
[139] Maldevel, 5 phases of penetration testing, 2016, URL https://securityblog.gr/3423/5-phases-of-penetration-testing/. (Online; Accessed 9 March 2020).
[140] J. Sunthonlap, P. Nguyen, Z. Ye, Intelligent device discovery in the internet of things-enabling the robot society, 2017, arXiv preprint arXiv:1712.08296. (Online; Accessed 9 March 2020).
[141] A. Acar, H. Aksu, A.S. Uluagac, M. Conti, A survey on homomorphic encryption schemes: Theory and implementation, 2017, CoRR, abs/1704.03578, URL http://arxiv.org/abs/1704.03578. (Online; Accessed 9 March 2020).
[142] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, M. Naor, Our data, ourselves: Privacy via distributed noise generation, in: S. Vaudenay (Ed.), Advances in Cryptology, Vol. 2006, EUROCRYPT 2006, Springer Berlin Heidelberg, Berlin, Heidelberg, 2006, pp. 486–503.
[143] W. He, X. Liu, H. Nguyen, K. Nahrstedt, T. Abdelzaher, PDA: Privacy-preserving data aggregation in wireless sensor networks, in: IEEE INFOCOM 2007-26th IEEE International Conference on Computer Communications, Vol. 2007, 2007, pp. 2045–2053, http://dx.doi.org/10.1109/INFCOM.2007.237.
[144] C.C. Aggarwal, P.S. Yu, A General Survey of Privacy-Preserving Data Mining Models and Algorithms, Vol. 2008, Springer US, Boston, MA, 2008, pp. 11–52, http://dx.doi.org/10.1007/978-0-387-70992-5_2.
[145] B. Zhou, J. Pei, W. Luk, A brief survey on anonymization techniques for privacy preserving publishing of social network data, SIGKDD Explor. Newsl. 10 (2) (2008) 12–22, http://dx.doi.org/10.1145/1540276.1540279, URL http://doi.acm.org/10.1145/1540276.1540279.
[146] O. Mazhelis, P. Tyrväinen, A framework for evaluating internet-of-things platforms: Application provider viewpoint, in: 2014 IEEE World Forum on Internet of Things, WF-IoT, 2014, 2014, pp. 147–152, http://dx.doi.org/10.1109/WF-IoT.2014.6803137.
[147] P. Sethi, S.R. Sarangi, Internet of things: Architectures, protocols, and applications, 2017, http://dx.doi.org/10.1155/2017/9324035. (Online; Accessed 9 March 2020).
[148] S. Vashi, J. Ram, J. Modi, S. Verma, C. Prakash, Internet of things (IoT): A vision, architectural elements, and security issues, in: 2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud), I-SMAC, 2017, 2017, pp. 492–496, http://dx.doi.org/10.1109/I-SMAC.2017.8058399.
[149] H. Derhamy, J. Eliasson, J. Delsing, P. Priller, A survey of commercial frameworks for the internet of things, in: 2015 IEEE 20th Conference on Emerging Technologies Factory Automation, ETFA, 2015, 2015, pp. 1–8, http://dx.doi.org/10.1109/ETFA.2015.7301661.
[150] D. Mendez, I. Papapanagiotou, B. Yang, Internet of things: Survey on security and privacy, 2020, arXiv:arXiv:1707.01879. (Online; Accessed 10 July 2020).
[151] T. Xu, J.B. Wendt, M. Potkonjak, Security of IoT systems: Design challenges and opportunities, in: 2014 IEEE/ACM International Conference on Computer-Aided Design, ICCAD, San Jose, CA, 2014, pp. 417–423, http://dx.doi.org/10.1109/ICCAD.2014.7001385.
[152] N. Zhang, S. Demetriou, X. Mi, W. Diao, K. Yuan, P. Zong, F. Qian, X. Wang, K. Chen, Y. Tian, C.A. Gunter, K. Zhang, P. Tague, Y. Lin, Understanding IoT security through the data crystal ball: Where we are now and where we are going to be, 2017, CoRR, abs/1703.09809, URL http://arxiv.org/abs/1703.09809.
[153] M. Ammar, G. Russello, B. Crispo, Internet of things: A survey on the security of IoT frameworks, J. Inf. Secur. Appl. (2018).
[154] M. Farooq, M. Waseem, A. Khairi, P. Sadia Mazhar, A critical analysis on the security concerns of internet of things (IoT), Int. J. Comput. Appl. 111 (2015) 1–6.
[155] Z.K. Zhang, M.C.Y. Cho, C.W. Wang, C.W. Hsu, C.K. Chen, S. Shieh, IoT security: Ongoing challenges and research opportunities, in: 2014 IEEE 7th International Conference on Service-Oriented Computing and Applications, Vol. 2014, 2014, pp. 230–234, http://dx.doi.org/10.1109/SOCA.2014.58.
[156] E. Fernandes, A. Rahmati, J. Jung, A. Prakash, Security implications of permission models in smart-home application frameworks, IEEE Secur. Priv. 15 (2) (2017) 24–30, http://dx.doi.org/10.1109/MSP.2017.43.

Leonardo Babun is a CyberCorps Scholarship for Service Alumnus. He is also a member of the Cyber-Physical Systems Security Lab (CSL) in the Department of Electrical and Computer Engineering at Florida International University. He completed his Doctoral degree in Electrical and Computer Engineering in 2020, and a Master degree in Computer Engineering in 2019, both from the Department of Electrical and Computer Engineering at Florida International University. He also completed a Master degree in Electrical Engineering at Florida International University in 2015. His research interests are focused on Cyber–Physical Systems (CPS) and the Internet of Things (IoT) security and privacy. Contact him at lbabu002@fiu.edu.

Kyle Denney is currently a Cybersecurity Researcher at MIT Lincoln Laboratory and a CyberCorps Scholarship of Service alumni. He previously completed his Master degree in Electrical Engineering in the Department of Electrical and Computer Engineering at Florida International University in 2019 and his B.S. in Applied Mathematics from Ferris State University in 2017. His research interests are focused on Cyber–Physical Systems (CPS) and hardware security of Internet of Things (IoT) and other lightweight devices.

Z. Berkay Celik is an Assistant Professor in the Department of Computer Science at Purdue University. His research investigates the design and evaluation of security for software and systems, specifically on emerging computing platforms and the complex environments in which they operate. Contact him at zcelik@purdue.edu.

Patrick McDaniel is the William L. Weiss Professor of Information and Communications Technology and Director of the Institute for Networking and Security Research in the School of Electrical Engineering and Computer Science at the Pennsylvania State University. Professor McDaniel is a Fellow of the IEEE, ACM and AAAS and the director of the NSF Frontier Center for Trustworthy Machine Learning. He also served as the program manager and lead scientist for the Army Research Laboratory's Cyber-Security Collaborative Research Alliance from 2013 to 2018. Patrick's research focuses on a wide range of topics in computer and network security and technical public policy. Prior to joining Penn State in 2004, he was a senior research staff member at AT&T Labs-Research. Contact him at mcdaniel@cse.psu.edu.

A. Selcuk Uluagac leads the Cyber-Physical Systems Security Lab at Florida International University, focusing on security and privacy of Internet of Things and Cyber–Physical Systems. He has a Ph.D. and M.S. from Georgia Institute of Technology, and M.S. from Carnegie Mellon University. In 2015, he received the US National Science Foundation CAREER award and US Air Force Office of Sponsored Research's Summer Faculty Fellowship, and in 2016, Summer Faculty Fellowship from University of Padova, Italy. Currently, He serves on the editorial boards of the IEEE Transactions on Mobile Computing, Elsevier Computer Networks, and the IEEE Communications and Surveys and Tutorials. Contact him at suluagac@fiu.edu.