symmetry

MDPI

Article

Developing Secure IoT Services: A Security-Oriented Review of IoT Platforms

Daniel Díaz López [1,*], María Blanco Uribe [1], Claudia Santiago Cely [1], Daniel Tarquino Murgueitio [1], Edwin Garcia Garcia [1], Pantaleone Nespoli [2] and Félix Gómez Mármol [2]

1 Faculty of Computer Science, Colombian School of Engineering Julio Garavito, Bogotá, 111166, Colombia; maria.blanco@mail.escuelaing.edu.co (M.B.U.); claudia.santiago@escuelaing.edu.co (C.S.C.); daniel.tarquino@mail.escuelaing.edu.co (D.T.M.); edwin.garcia-ga@mail.escuelaing.edu.co (E.G.G.)

2 Faculty of Computer Science, University of Murcia, 30100 Murcia, Spain; pantaleone.nespoli@um.es (P.N.); felixgm@um.es (F.G.M.)

* Correspondence: daniel.diaz@escuelaing.edu.co; Tel.: +571-6683600-262

Received: 11 October 2018; Accepted: 30 October 2018; Published: 27 November 2018

Abstract: Undoubtedly, the adoption of the Internet of Things (IoT) paradigm has impacted on our every-day life, surrounding us with *smart* objects. Thus, the potentialities of this new market attracted the industry, so that many enterprises developed their own IoT platforms aiming at helping IoT services' developers. In the multitude of possible platforms, selecting the most suitable to implement a specific service is not straightforward, especially from a security perspective. This paper analyzes some of the most prominent proposals in the IoT platforms market-place, performing an in-depth security comparison using five common criteria. These criteria are detailed in sub-criteria, so that they can be used as a baseline for the development of a secure IoT service. Leveraging the knowledge gathered from our in-depth study, both researchers and developers may select the IoT platform which best fits their needs. Additionally, an IoT service for monitoring commercial flights is implemented in two previously analyzed IoT platforms, giving an adequate detail level to represent a solid guideline for future IoT developers.

Keywords: Internet of Things; IoT platforms; information security; IoT security

1. Introduction

The Internet of Things (IoT) has arisen as an emerging technology with notable expansion potentialities during the last decade [1]. IoT envisions a future where billions of smart objects will communicate autonomously among them to ultimately provide services to humans [2]. In particular, IoT architectures are composed of an interconnected network of intelligent devices, which take advantage of Internet connectivity to collect, exchange and process data [3]. These devices may be located in home appliances, buildings, vehicles or monitoring infrastructures, and they may be remotely controlled by software services that allow them to be managed [4,5]. Thus, everyday "things" may be controlled by already existing network infrastructures. In this way, it is possible to create a more concrete integration between real world and computer-based systems, with great benefits, both from an economic viewpoint and for the final users [6,7]. Specifically, under the IoT paradigm some crucial application domains will be enhanced, such as health-care, environmental and industrial plant monitoring, etc. [8]. In particular, the health-care ecosystem is heavily profiting from the integration with the smart objects, achieving notable results in the patients' care [9,10]. To better perceive the IoT potentialities, in 2016, Gartner estimated 6 billion devices were connected, forecasting an additional 21 billion smart things by 2020 [1]. Additionally, a prevision of Cisco System reveals that

IoT industry will create $14.4 trillion considering the combination of increased revenues and lower costs for companies between 2013 and 2022 [11].

Following this reasoning, the potential benefit derived from the IoT ecosystems has attracted attention of both academia and industry, which foresees clear market advantages in adopting IoT solutions [12]. Consequently, the popularity of IoT services is certainly increasing [13], fostering as well intensive research on this new field. Nonetheless, developing applications in the IoT context may represent a cumbersome task, mainly due to the following challenges:

- Lack of exhaustive documentation of the existing frameworks
- Multiple programming languages to control the devices
- Hardware heterogeneity of the devices
- Resource constraints inherent to the IoT nodes
- Several communication protocols used by the devices

Consequently, designers and developers bare the burden of managing the entire IoT infrastructure and handling both software and hardware layers of the platforms. Additionally, resource management and distribution within the network have been proved as cumbersome tasks, which need specific solutions to address them [14,15]. Thus, to tackle the aforementioned challenges, several big players decided to propose their own IoT frameworks to support the development of IoT services. On the one hand, having several choices to develop an IoT application surely represents an advantage. On the other hand, the proposed frameworks present several peculiarities which make them more suitable for certain applications.

In this context, a deep study of the security features exposed by some of the most relevant IoT platforms over which IoT services are being implemented is needed to guarantee that data are being managed with confidentiality, integrity and availability. Specifically, it has to be clarified to which extent the existing threats targeting IoT ecosystems may impact on the analyzed frameworks, to argue on their risks and on possible countermeasures [16].

The main goal of this research work is to concretely evaluate the most prominent commercial proposals in the context of IoT frameworks from a security viewpoint, in order to built a detailed guideline for worldwide researchers. By doing so, scientists aiming to exhaustively develop and test their IoT prototypes may choose which platform fits better to their needs, thus decreasing the total development time for their research works. Additionally, IoT service developers may also use this work as a base for selecting the correct platform to deploy solid and secure applications. This work evaluates existing IoT frameworks based on five common criteria (as detailed in Section 3) and uses them as the basis for comparing their security features. The goal of our analysis is to highlight advantages and disadvantages of the presented frameworks, so that one could safely choose an IoT platform in order to implement a service.

Selecting the IoT platforms for conducting security analysis is not straightforward due to the enormous number of companies offering platforms or components required to deploy IoT services. Both academia and industry have suggested reports summarizing remarkable IoT platforms, aiming to provide a solid guideline. Among them, Gartner published the "*Competitive Landscape of IoT Platform Vendors*" (https://www.gartner.com/doc/3730917/competitive-landscape-iot-platform-vendors) which reviewed 10 IoT platforms considered as emergent in the IoT market. Moreover, Postscapes, which has recently become a reference of trends for IoT stakeholders, published a review called "*IoT Cloud Platform Landscape*" (https://www.postscapes.com/internet-of-things-platforms/). This report indexed more than 123 outstanding platforms, including proprietary and open source solutions, which are directed to different target audience.

As a result of reviewing many reports, such as the ones mentioned previously, we understand the necessity of having a broader view about competitors in the IoT platform ecosystem. Consequently, we have selected a subset of platforms for our security analysis, including traditional big market players as well as platforms proposed from new ones. For the latter, we believe they can become strong IoT providers in the next future because of their proposed IoT functionalities and vision.

Overall, in this work, we review the following platforms: (i) *Samsung Artik*; (ii) *Amazon IoT*; (iii) *IBM Watson IoT*; (iv) *Oracle IoT*; (v) *Evrythng*; (vi) *Dweet*; (vii) *Node-Red*; (viii) *Nimbits*; (ix) *The thing system*; and (x) *Sitewhere*. The main characteristics of these platforms are highlighted, and then they are compared against the selected criteria to argue on their main advantages and disadvantages. Furthermore, this work provides a detailed description of an IoT avionic service, together with its implementation on two of the analyzed platforms. Going through the suggested guidelines, readers may build a solid ground and acquire the required knowledge to start developing their own prototype based on the suggested examples.

The reminder of this article is structured as follows: first, a comparison with the state-of-the-art in the field is given in Section 2. Then, Section 3 defines the security criteria that should be considered within an IoT platform. Section 4 introduces the concept of IoT service, and briefly introduces the IoT platforms on which they can be implemented. A deep analysis of the major IoT platforms available on the market is given in Section 5, comparing them with respect to the identified security criteria. Section 6 describes an IoT scenario based on an avionics service, focusing on the IoT devices and rules. Then, Section 7 details the steps to implement the avionics scenario in two selected IoT platforms. Finally, relevant conclusions and possible future works are described in Section 8.

2. Related Works

As previously stated, the IoT technologies are in continuous expansion. Thus, several research papers have been recently published aiming at surveying the IoT domain from different perspectives. In this context, in [17], a survey on enabling technologies, protocols and application issues is presented. The main objective of this work is to provide an outline on the above-mentioned challenges to endow researchers and developers with a quick overview on how the different protocols can be combined to deliver IoT solutions without going deeply in the protocols' details. Similarly, authors in [18], after surveying the enabling technologies, proposed key IoT applications which may benefit industries, such as health-care, food supply chain and many others.

Besides the cited works, researchers focused their attention not only on reviewing the proposals presented by the academic community, but also on analyzing the commercial IoT solutions brought to the market by industrial organizations. The main reasoning behind this choice is that understanding how the industry utilizes technologies in the IoT landscape may be crucial for researchers to identify trends, opportunities and open challenges. To this extent, authors in [19] analyzed several IoT solutions in the marketplace in order to argue on the potential applications and the technologies used. Specifically, they identified trends in the industry-based IoT solutions, classifying them under five different categories, namely: smart wearable, smart home, smart city, smart environment, and smart enterprise. Furthermore, Ganguly [20] compared cloud IoT platforms based on four factors: technical offerings, strategy, market presence, and compliance. Each factor includes sub-factors, thus the author could conclude with generic claims on the benefits of adopting a specific platform. During the study, two security features are analyzed as a sub-factor of technical offerings, that is message level security and data encryption. Apparently, the author assumed that the transport layer security is already present. Additionally, authors in [21] compared four IoT platforms (three open-source and one commercial) against a reference architecture. By doing so, the authors claimed that the proposed architecture achieves an adequate abstraction level so that it is possible to ease the comparison of different platforms. Another comparison is presented in [22], where several commercial IoT frameworks and platforms are analyzed. The authors provided a comparative study based on the following selected criteria: architectural approaches, supported protocols, interoperability, security, hardware requirements, governance and support for application development. Seventeen frameworks are considered, giving suggestions to identify the most suitable framework to develop future projects.

Despite their contributions, the above-mentioned works neglect (totally or partially) the security features of the selected IoT platforms. One could safely argue that an in-depth security review constitutes a need for the market-place proposals, since the IoT ecosystem is still surrounded by major security challenges [23]. We believe that researchers and developers must be aware of the security risks to which they are exposed while developing their own IoT applications. Recently, the authors in [24] reviewed a subset of industrial frameworks focusing on their security characteristics. In particular, the authors analyzed authentication, access control, secure communication support and cryptography primitives, thus giving an overall picture of the security level of the selected platforms. Nevertheless, the selected criteria look quite generic from a security viewpoint. That is, the authors did not consider the communication among all the involved entities of the architecture (users, devices, and framework). To this extent, more detailed criteria are required, specifying how the different entities are associated with security measures within the analyzed architecture. Additionally, the authors neglected the anomaly detection in the security analysis. One could argue that the capability of detecting anomalies is of primary importance in a full-fledged IoT framework, so that the security incidents can be reported in a timely fashion. Finally, the work in [24] does not present a concrete implementation of an IoT service on the surveyed platforms.

To address these limitations, in this work, we propose the implementation of flight monitoring service using two analyzed platforms. The motivation behind this choice is two-fold: On the one hand, IoT service developers may use the proposed guidelines as a base to ease the implementation of their own proposal. On the other hand, developing the avionic IoT service on the above-mentioned platforms helps us to better argue on their security features.

3. Security Criteria for Comparison

Thus far, several IoT platforms have been proposed by different players aiming at helping the development of IoT services. These platforms share a common ground, but also exhibit considerable dissimilarities in terms of architecture and concerning how the services can be implemented. Additionally, they expose significantly different security features with respect to the data management and the communication among the involved entities (users, devices and platform itself) [25]. To this end, after analyzing various IoT platforms, we have identified five common security criteria which may be useful for our comparison. It has to be stated that not all the examined platforms fulfill the identified criteria, but we believe that their presence surely represents an added value from a security viewpoint. The aforementioned reasoning led us to choose the following security criteria:

1. **Authentication**: The authentication process provides a way of identifying an entity within the system. In the context of IoT, the produced data are massive, causing several security and privacy issues, especially regarding the authentication among the devices, the users and the system itself [26]. To address this limitation, various authentication schema have been proposed [27,28]. Thus, we considered the following three sub-criteria:
    - Authentication protocol between IoT devices and IoT platform
    - Authentication protocol between the IoT platform and users (both end-users or administrators)
    - Authentication protocol among the components of the IoT platform
2. **Encrypted information management**: The encryption process takes care of encoding a message or an information so that only authorized entities can access it [29]. Due to the amount of information and the resource-constrained nature of the IoT devices, the encryption assumes a fundamental role. In this case, we analyzed the encryption at two levels:
    - Encryption for data at rest (i.e., data physically stored and inactive)
    - Encryption for data in transit (i.e., data transmitted among the entities)
3. **Authorization**: Once a user has been successfully authenticated in the system, the authorization process determines whether the user has the rights to access a given resource or to execute an activity. As in the case of authentication, authorization strategies represent a strong requisite in the IoT ecosystem, since unauthorized intruders can perform malevolent actions, such as compromising the integrity of the system by maliciously modifying its data [30]. We detailed this feature in two sub-criteria, as follows:

    - User authorization to perform operations within the IoT platform
    - Authorization of the IoT devices to perform operations among the components of the IoT platform

4. **Accounting**: It measures the amount of resources a specific user consumes during their access. Examples of resources include the session time or the data which the user has sent and/or received during a session [31]. Given the huge number of performed operations within the IoT platform, this parameter is also crucial from a security perspective. In addition, in this case, we extracted the following sub-criteria:
    - Accounting for operations that users perform over the data and IoT devices (e.g., read, write, aggregation, etc.)
    - Accounting for operations created for IoT services components on data and IoT devices (e.g., accounting, disassociation, update, etc.)
5. **Anomaly detection**: It refers to the ability of the system to spot anomalies among the normal activities, which may indicate the presence of a security incident [32]. In this context, this criterion is related to the IoT platform capability to detect anomalies in IoT service's state or in the normal operation of its components [33]. We consider this characteristic of primary importance in a full-fledged IoT framework, so that the security incidents can be reported in a timely fashion, and possible countermeasures can be undertaken [16].

These five security criteria are essential to estimate the negative impact that a security incident can provoke in an IoT service. To estimate the impact is paramount to understand that an IoT service can be internally composed of a set of communication protocols at different layers (application, transport, network, access control and physical), each one of them posing some known or undiscovered vulnerabilities [25]. Additionally, this IoT service can also be represented through an architecture of components (perception or device, network or transmission, middleware, application and business) that will define the attack surface of the service. A security incident (damages to the confidentiality, integrity or availability) can affect any IoT protocol or component causing an impact that can be categorized as reputational, operative or legal. The reputational impact refers to the damage of the image of the IoT service producing consequences such as the loss of IoT user trust or massive unsubscriptions. The operative impact refers to the activities that must be developed as a consequence of an incident which has a cost represented in the time of the employees working to operate the IoT service or the hiring of additional specialized personnel. Finally, the legal impact refers to legal issues for the IoT service provider due to the non-compliance of regulations, for example laws for user data protection or laws for the operation of services which can be considered of high criticality.

4. IoT Platforms Overview

According to the Boston Consulting Group market analysis, the Internet of Things market expects to reach $267B by 2020 spent in IoT technologies, products and services [34]. Additionally, by 2025 the number of intelligent *things* will grow to 75.4 billion of installed devices: (i) impacting the automation by connecting machines, sensors and actuators to automate industrial processes; (ii) integrating the data from a machine or sensor with the organizational databases and other data sources such as open government databases, social media feeds, etc.; and (iii) moving to a service-oriented business model through the above-mentioned automation and integration processes.

Globally, companies increasingly care about how the IoT technology could fit in their architectures and how the generated flows of data may be integrated within the existing data sources. Therefore, big players such as Amazon Web Services, Cisco, IBM, Ericsson and others converged on the strategy of offering IoT services and platforms. These platforms are "cloud-based software packages and related services that enable and support sophisticated IoT services" [35], enabling enterprises to manage millions of devices and collect information to perform big data analysis, secure their information and offer professional services. Moreover, the offered IoT services could be applied and thus enhance several fields such as industries, government, health, telecommunications, transportation, home automation, security and so on. Next, we present some of the most prominent IoT platforms nowadays available for developers. This selection was driven by the comparison criteria previously described in Section 3.

4.1. Samsung Artik

Samsung Artik [36] is an IoT platform that integrates and provides interoperability between IoT devices, cloud repository, Apps and services. The platform provides APIs, SDKs and tools to the developers to interact with the IoT components and the cloud using REST/HTTP to easily build Apps and services. It supports wired and wireless protocols (Ethernet, Wi-Fi, Bluetooth Smart, ZigBee, Thread, etc.) and collects heterogeneous data both from the devices and the cloud services. The platform has capacities of aggregation, analytics and visualization of data in near real time [37].

4.2. Amazon IoT (AWS IoT)

Amazon IoT or AWS IoT [38] is a cloud-based platform that enables the interaction between the connected devices and the AWS Services and other devices. The devices connect to the platform using a gateway, which is responsible of managing all the active devices connections and permits a bidirectional communication, low latency and scalability. The IoT devices can exchange messages with AWS IoT platform using Message Queue Telemetry Transport (MQTT), HTTP or WebSockets protocols through the AWS IoT Device SDK. The Amazon IoT platform provides mutual authentication, secure data exchange, authorization control based on X.509 certificates, tokens and registry. Furthermore, AWS IoT has a rules engine that allows building IoT applications that collect, process, analyze and act on data generated by IoT devices [39].

4.3. IBM Watson IoT

IBM Watson IoT Platform [40] is a cloud-hosted managed service designed to simplify the extraction of values from the IoT devices. This platform is able to register, connect and manage any sensors or mobile devices connected to the Internet. Additionally, it supervises the connectivity between devices and the platform, and it shows device data in real-time to define rules that may trigger alerts or other actions. To communicate with the IoT devices, applications, third-party services, data storage and users, Watson IoT provides an API, together with device management agents and messaging protocols like MQTTv3.1, MQTTv3.1.1 and HTTP. Within the platform, the analytics services (i.e., data visualization in real-time and edge and cloud analytics) are visualized through a user-friendly dashboard. Finally, the architecture of the Watson IoT Platform offers secure connectivity through authentication services using client ID, authentication tokens, certificates and Transport Layer Security (TLS) [41].

4.4. Oracle IoT

Oracle IoT [42] is an integrated IoT platform that allows private communication between IoT devices and the Oracle cloud services through Java components (i.e., Java, Java Card and Java Embedded) and Berkeley DB technologies, offering specific Oracle solutions like Big Data Analytics, Big Data Management and Oracle Fusion Middleware. Specifically, the platform allows end-to-end security, integration with enterprise IT systems, like Enterprise Resource Planning (ERP) and Customer Relationship Management (CRM), secure data management and analysis. The platform visualizes the IoT devices in order to facilitate and standardize their integration with the enterprise IT infrastructure and systems. For a developer team, Oracle IoT provides a single Java base platform for the application development that decreases constructions and deployment times. Additionally, it provides secure integration with Oracle and non-Oracle applications and services and IoT devices using REST API [43].

4.5. Evrythng

Evrythng [44] is an IoT platform used to evolve traditional products into smart products. It allows the customer to access key product information, enabling also the interaction with such information through a mobile web application. Manufacturers or retailers can also use the platform to observe insight relations between consumer and products, monitor the supply chain and control the retail process. Further, Evrythng offers a digitalization of products, which is supported by different components, such as: integration with platforms, e.g., ERP, CRM or Content Management System (CMS); administration and analytics; real-time data management; security and access control; product connection management (to connect with smart products); and developer tools (to offer applications for costumer and manufacturers). This platform is used by companies such as Coca-Cola, General Electric, Google, and others [45].

4.6. Dweet

Dweet [46] is a platform that is enabled to receive data (called *dweets*, i.e., a “Twitter for things”) from IoT devices through a HAPI (Humanized web API) or an API console. Using Dweet, it is also possible to request data (dweets) from an IoT device. To this extent, persistence and privacy of data received from the device can be defined according to the user subscription. Moreover, alerts regarding the values can also be defined. Dweets can have a payload of up to 2000 characters and it is developed by Bug Labs, a company that also offers another product called Freeboard [47]. Freeboard is a web service that can be integrated with Dweet which allows to visualize the values reported by several connected devices. Dweet is used by companies such as Ford, Verizon, Renesas and others.

4.7. Node-Red

Node-Red [48] is a programming open source tool for interconnecting hardware, APIs and online services. In Node-Red it is possible to define flows between more than 1392 existing nodes or new defined nodes. It is built on Node.js and can be run locally in devices such as Raspberry Pi, BeagleBone Black, Android based devices or Arduino, or in the cloud environments through IBM Bluemix, Amazon Web Services, Microsoft Azure or SenseTecnic FRED. The flows among the nodes are stored in JSON format which enables them to be shared. Additionally, the different nodes in Node-Red are able to cypher the information and generate alarms according to the received information. Node-Red is used by companies such as Agilit-e, Sense Tecnic Systems, Spirit AI and others [45,49].

4.8. Nimbits

Nimbits [50,51] is a PaaS platform that can be used to develop software and hardware solutions that are easy to connect to each other. Using Nimbits, it is possible to log events reported by IoT devices and to react to those events in real time. Besides, it can be installed into a Raspberry Pi or in a robust cloud environment (e.g., Google App Engine), being possible to use different clients (Java, Javascript, Android) to connect to it. A subscription system that sends log messages, alerts and events to the user is also supported. Nimbits data are stored using a time based algorithm which allows to store all the data in an uncompressed time stamp way to guarantee a quick read access. Unfortunately, this platform was recently deprecated (July 2018), after we had finished our comparative work, however we decided to include it because of its 17-year life-cycle and the security functions that it reached during its existence even being an open source project.

4.9. The Thing System

The Thing system [52,53] has an architecture based on a central component called “steward” which makes all the interactions with clients, browsers and devices (from different types). Steward is composed of different modules, being the most important ones a rule engine, a database, a web server, an API provider, a set of prototypes (with its corresponding drivers), a discovery module and a connector to native protocols. With The Thing system, it is possible to connect regular devices to the cloud so these can be monitored and controlled. Steward is written in `Node.js` and can run on servers or fit onto a small single board computer such as the Raspberry Pi. The platform can connect to devices through WiFi, ZigBee, Z-Wave, USB or Bluetooth.

4.10. Sitewhere

Sitewhere allows connecting devices with MQTT, Advanced Message Queuing Protocol (AMQP), Simple Text Orientated Messaging Protocol (STOMP) and other protocols, by means of which the full life cycle of registration, sending of commands, recollection of data and aggregation, is supported. Sitewhere infrastructure is composed of a Sitewhere server, a database (MongoDB, Apache HBase or InfluxDB) and a MQTT broker, and it can be installed on premise or in a cloud based environment (Microsoft Azure or Amazon EC2). Sitewhere leverages different technologies, e.g., Apache Tomcat, Spring Framework, Spring Security and Hazelcast. Sitewhere allows multiple customers to run over a single Sitewhere instance, each with their own data store and processes. API Rest Services are also offered by Sitewhere so interaction with data from a third party is also possible [54,55].

5. IoT Platforms Analysis

In this section, we evaluate the five security criteria defined in Section 3 for the platforms introduced in Section 4.

5.1. Samsung Artik

5.1.1. Authentication

From IoT devices to the cloud, a device must have an OAuth2 access token to make a registration process, i.e., the IoT device authentication with the cloud. This device token is obtained from Artik Cloud using one of two possible methods: (i) sending a PUT call to the Samsung API REST sending the device ID and receiving from the cloud the device token as response; or (ii) asking for the generation of the device token from the Artik Cloud interface directly. In both cases, the device token is loaded in conjunction with the device ID in the IoT device, so the latter can register with the Samsung cloud. A device token does not have a default expiration time, but it can be revoked.

Between the users and the cloud, Samsung Artik offers authentication using a sign in interface that allows to use an Artik Cloud Account or other identity providers. The user token is returned as a result of a successful sign in process.

Between components in the cloud, applications must get an application token to execute API calls. With a valid token, an application can access the user data, if the user has previously authorized it. An application token cannot be refreshed, therefore the use of the Client Credential method must be used to get a new token.

5.1.2. Encryption

As for data at move, Artik supports the use of DTLS to secure the CoAP protocol, requiring that all communications are encrypted using a server certificate. The CoaP Server certificate hosted by the cloud is verified by the IoT device using an intermediate certificate lying in a trust store. Artik also supports the use of the MQTT protocol to send data messages or receive actions; however, they must use an encrypted channel. Additionally, Artik supports REST and WebSockets (API connections for transferring messages) which also require a previous encrypted channel. This requires that the IoT device supports SSL.

For data at rest, Artik defines the Samsung Artik security architecture where data protection at final devices and in the cloud service is provided. Thus, Artik defined a TrustZone, unique keys for encryption and eMMC (embedded MultiMediaCard) secure file system to secure storage.

5.1.3. Authorization

The cloud uses OAuth2 access tokens to authenticate users. User tokens can be obtained using one of two possible methods:

- Authorization code: Once the user signs using an authentication method, she is redirected to the IoT application with an authorization code which at the same time allows the application to ask for an application token used to make API calls.
- Implicit method: Once the user signs using an authentication method, she is redirected to the application server along with a token that is used by the application to make API calls. Only an application with a token can make API calls. Artik also permits sharing devices through an invitation sent by email, which allows the user to see the device data and send actions on the device.

Between cloud components, Samsung allows defining whether an application created in the cloud can read or write over a specific IoT device.

5.1.4. Accounting

Accounting of operations from the users to the data or the IoT devices is done by recording the date and time when a device sends data to the cloud. Accounting also tracks when a device is created. However, there is no apparent accounting of the actions that are sent to a device by a user.

Regarding accounting of operations from the components to the data or the IoT devices, there is not information available about this security feature.

5.1.5. Anomaly Detection

It is possible to use Anodot (real time analytics and automated anomaly detection system) to detect anomalies in the Artik traffic. Anodot uses machine learning algorithms to characterize the normal behavior of the data stream and sends an alert when there is abnormal data in the IoT ecosystem. Alternatively, it is possible to use Machine Learning APIs to train a model with data from devices to predict abnormal inputs and also define rules that include a prediction or anomaly detection condition and an operator to match the result of the prediction or anomaly detection.

In conclusion, Samsung Artik provides mechanisms for authentication and authorization between devices, users, components and the cloud using tokens and data privacy using cryptographic protocols for network transmission and storage. On the other hand, the platform provides accounting services through the registration of some device operations such as its creation and the data transmission. Finally, anomaly detection is present in Samsung Artik using Machine Learning API calls and Anodot.

5.2. Amazon IoT (AWS IoT)

5.2.1. Authentication

From IoT devices to the cloud, each connected device must have a credential (X.509 certificates, IAM users and groups, or Amazon Cognito identities) to access the message broker or to use the AWS IoT Thing Shadows service (this service maintains JSON documents with information about device state).

Between users and cloud, users have accounts to log in the AWS IoT Management console, which are accessed using credentials. For web applications, desktop-type applications and CLI commands Amazon IoT provides the Identity and Access Management (IAM). IAM is a web service to control the access of the users to AWS resources.

Between cloud components, the message broker (publish/subscribe broker service that enables the sending and receiving of messages to and from AWS IoT has the responsibility to provide a secure mechanism to send and receive messages between devices and AWS IoT application using MQTT protocol or MQTT over WebSocket protocol. Each device has a default MQTT client ID and it is possible to use the HTTP REST interface to publish messages.

5.2.2. Encryption

Regarding data at move encryption, Transport Layer Security (TLS) is used to ensure the confidentiality of the application protocols (MQTT and HTTP) and all the traffic to and from the platform. AWS cloud security mechanisms protect data as they move between AWS IoT and other devices or AWS services.

Encryption of data at rest with AWS IoT is possible to save files into Amazon S3 (Amazon Simple Storage Service) and Server-Side Encryption service (SSE-S3) allows protecting data at rest with multi-factor encryption using AES-256 (256-bit Advanced Encryption Standard). Besides, certificates X.509 and asymmetric encryption are used as a mechanism to secure storage on the devices.

5.2.3. Authorization

For operations on the IoT platform, policies determine what an authenticated identity can do. The entity (devices, mobile application, web application or desktop application) can execute AWS IoT operations only if it has a policy that grants the permission. An AWS IoT policy controls what a device, user, or application can do in AWS IoT. An AWS IoT policy is a JSON document that conveys one or more policy statements containing an Effect, an Action, and a Resource. The Effect specifies whether the action will be allowed or denied. The Action specifies the action that the policy is allowing or denying. The Resource specifies the resource or resources on which the action is allowed or denied.

With regards to authorization of operations between components of the IoT platform, after a comprehensive review of the platform, we did not find information available about this security feature.

5.2.4. Accounting

As for accounting of operations from the users to the data or the IoT device, AWS IoT sends information about user authentication (success and errors) and authorization to AWS IoT CloudWatch monitoring service. Additionally, AWS IoT is integrated with CloudTrail. The CloudTrail log files (JSON-formatted events) contain logs about all AWS IoT actions (API Calls and delivers) to obtain information about the request that was made to AWS IoT: for example, the source IP address from which the request was made, who made the request, when it was made, and so on.

After comprehensive review of the platform, we did not find information available about accounting of operations from the components to the data or the IoT devices.

5.2.5. Anomaly Detection

AWS IoT provides various tools to monitor itself. Amazon IoT has a component known as AWS Lambda, which detects anomalies in the IoT platform. It is possible to collect monitoring data from the AWS solution and debug a multi-point failure if one occurs through a monitoring plan that was defined previously and a baseline for normal AWS IoT operation. Moreover, AWS provides several automatic and manual tools for monitoring AWS IoT. Automated tools can be used to monitor the AWS IoT and report whether something is wrong. These automated tools are: Amazon CloudWatch Alarms, Amazon CloudWatch Logs, Amazon CloudWatch Events and AWS CloudTrail Log Monitoring. Manual tools can be configured for those situation that the CloudWatch Alarms does not cover and particular IoT solutions need.

In conclusion, AWS uses credentials, certificates and supports platforms for the authentication process. It makes use of secure protocols for the transfer of data between components and symmetric encryption such as AES for data storage. For authorization, it has policies and identity control in the different elements of the systems to obtain privileges in the IoT platform. Finally, AWS IoT offers mechanisms to monitor user authentication and authorization activities and has tools for monitoring operations in the system in order to detect anomalous situations.

5.3. IBM Watson IoT

5.3.1. Authentication

For IoT to cloud device, the devices can be registered to the cloud or generate an API Key. In any case, the devices have unique credentials that identify them against the platform.

Between users and the cloud, the browser-based GUI and REST APIs are fronted by HTTPS and to ensure the identity, Watson IoT Platform uses certificate signed by DigiCert. The user can use ID/password authentications, OAuth token authentication and client certificate authentication. The cloud services support authentication through LDAP and IBM web identity. Finally, the platform can use the IBM Single Sing On (SSO) service to implement user authentication.

5.3.2. Encryption

Regarding data at move, it is implemented in the IBM Watson platform using MQTT and TLS to encrypt the channel. On the other hand, data are exchanged from client to server through the PPTP (Point-to-Point Tunneling Protocol), SSL or IPSec VPN gateways. Furthermore, the Watson platform allows AES symmetric encryption and RSA asymmetric encryption.

Regarding data at rest, the connection between the applications and the database is protected by SSL certificates that DigiCert signs. Here, the encryption of user data stored on the platform and the devices is the responsibility of the developers of the applications and the IBM platform has data-related services available in its catalog to help with these concerns.

5.3.3. Authorization

Referring to operations on the IoT platform, IBM Watson uses security polices, blacklists and white-lists to specify which devices are allowed to connect to the cloud. On the other hand, it provides a device management API to disable their direct connection to the platform. Besides, the platform uses security polices to accept or deny privilege to the users.

As for authorization of operations between IoT platform components, the platform defines a topic space for each single organization for a client authenticated, it contain devices and application and being impossible to access data from one topic space to another. This means that each application is isolated and runs in its own container that has specific resource limits. At the same time, there are client-provided authentication credentials, which dictate which devices can interact in the IBM Watson IoT spaces, preventing one device from impersonating another device.

5.3.4. Accounting

Each user authentication attempt in the developed application generates an input in the Data operations log.

Regarding accounting of operations from the IoT devices, the system registers information about device that are violating security policies. Such devices can notify to the Watson IoT platform device management support of their changes using the Device Management Protocol. The notifications generate a new log entry that includes a log message, time stamp, severity, and optional base64-encoded binary diagnostic data.

5.3.5. Anomaly Detection

Protection against intruders is applied in order to discover threats that can be solved. Finally, Bluemix (a cloud platform as a service developed by IBM) uses the Tenable Network Security vulnerability scanning tool, Nessus, to detect any problems in the network and host configuration in order to solve possible problems.

In conclusion, IBM Watson provides authentication services using credentials and certificates to user, devices and components. The platform protects the data at move and at rest through symmetric and asymmetric encryption, secure protocols and data-related services available in the IBM Cloud catalog but also clarifies that the data protection in rest depends on the application. On the other hand, for authorization, it is possible to implement security policies, black/white list and topic spaces. Finally, the platform implements mechanisms to account for user operations and devices events, as well as threats analysis processes.

5.4. Oracle IoT

5.4.1. Authentication

The Authentication from IoT devices to the cloud is done through a process that involves registration and activation. In the registration process, it is possible to define an activation secret value in the Oracle IoT Cloud, which must also be loaded in the IoT device. After activation, an OAuth access token is assigned to the device which expire after 1 h by default.

Regarding authentication between the users and the cloud, users have an account to log in the Oracle IoT Cloud, which is accessed using credentials (User/password).

For authentication between cloud components, Oracle IoT Cloud uses a Certificate Authority (CA) certification to enable REST clients to connect securely to the Oracle IoT Cloud Service. Most REST requests use HTTP basic authentication, even if some of them use OAuth for authentication. Enterprise applications must register with Oracle IoT Cloud Service and obtain an ID and Shared Secret to obtain appropriate credentials for calling Oracle IoT Cloud Service REST API resources. The Oracle IoT Cloud contains a Code Trust store which hosts code security certificates to allow software authentication.

5.4.2. Encryption

As for the encryption of data at move, the devices that connect directly to the Oracle IoT Cloud use client Software Libraries that allow them to support security functions. The communication protocol between devices and Oracle IoT Cloud is HTTPS over TCP/IP. Communication between REST consumer applications and the Oracle IoT cloud can be made using JSON Web Token (JWT) (RFC 7519) that defines a compact and self-contained way for securely transmitting information between parties as a JSON object. This information can be verified and trusted because it is digitally signed. JWTs can be signed using a secret (with HMAC algorithm) or a public/private key pair using RSA.

In the case of encryption of data at rest, the devices can use a Trusted Assets Manager (TAM) API which provides methods for encryption using a shared secret and a private key. Depending on the capability of the device, it is possible to store sensitive trust material in a plain persistent store file system, or in a hardware keystore, such as a pluggable authentication module (PAM), or in a secure token.

5.4.3. Authorization

About authorization of operations over the IoT platform, Oracle IoT Cloud uses roles (Administrator, Operator and User) and user accounts to control access to the available features and resources. Each role has different privileges to execute operations over the IoT platform. Operations are mainly querying data, sending commands/messages to the devices and subscribing data. Additionally, Oracle IoT Cloud uses CORS (Cross-Origin Resource Sharing) to specify whether the user browser can access a resource or execute an operation (from the API REST) in the Oracle cloud.

For authorization of operations between components of the IoT platform, Oracle IoT Cloud uses an OAuth2 access token. This is a supported authorization option and is required for certain resources, such as REST API messages. The token must first be obtained by authenticating with the OAuth2 server included with Oracle IoT Cloud Service.

5.4.4. Accounting

With regards to accounting of operations from the users to the data or the IoT devices, there is no information available about this security feature.

Regarding accounting of operations from the components to the data or the IoT devices, it is possible to monitor the links activity and analytics processors. Oracle IoT Cloud has a REST API which allows obtaining device activation logs identified by device ID. Additionally, it is possible to view logs of the application and Analytics Logs to monitor the activity or troubleshoot problems using different options of Log Type and Log Container.

5.4.5. Anomaly Detection

After a comprehensive review of the platform, we did not find information available about this security feature. In conclusion, Oracle IoT provides authentication services by devices registration, user credentials and certificates for connection between components. Likewise, it has utilities that allow encrypting data that travel through the network and that are stored in devices and the cloud, as well as secure protocols for the information transmission. On the other hand, it allows the use of OAuth2, user roles and REST API requests for authentication. Finally, it allows monitoring activities from devices and applications.

5.5. Evrythng

5.5.1. Authentication

From IoT devices to the cloud, IoT devices are authenticated using secret and unique keys generated by an API. Each request sent to the cloud is authenticated by a device key. In this way, only devices that have been previously registered and hold a key can connect to the cloud and are able to access the IoT platform. IoT devices connecting to the platform also have the possibility of using OAuth to make the authentication leveraging a trusted server such as Facebook, Twitter or Google.

Authentication between users and the cloud is also considered as Evrythng dashboard implements two-factor authentication and a Time-based One-time Passwords (TOTP). Users connecting to the platform can also use OAuth to make the authentication through a trusted server as Facebook, Twitter or Google.

Authentication between cloud components is done using a Public Key Infrastructure (PKI). The public keys of the servers are stored previously in the Software Development Kits (SDK) so the initial key transfer is secure. Mobile and desktops connecting to the cloud have a root certificate in their trust store. All components must be authenticated to be able to start a communication.

5.5.2. Encryption

Data at move are protected with encryption, since all the communications toward the IoT platform coming from a device, an application or a browser are sent using TLS. AES (Advances Encryption Standard) is used as the encryption mechanism. Additionally, communications between component in case are also ciphered.

Data at rest are also protected; specifically, data resting in the cloud are stored in disk volumes which are encrypted using a key management infrastructure. In addition, a cluster of NoSQL databases is used to guarantee redundancy in the event that one of the databases becomes corrupted or inaccessible.

5.5.3. Authorization

The authorization operations over the IoT platform and between the IoT platform components are: access, manage and share. These operations can be designated even over a single property of a device, and can be assigned to a device, user or system. This allows defining a scenario where data are shared between different IoT components.

5.5.4. Accounting

Accounting of operations from the users to the data or the IoT device is done through the Evrythng platform who registers all the requested operations originated from an application or a user. There is a centralized log mechanism allowing an audit over the access which would lead to the detection of some suspicious unauthorized actions.

Accounting of operations from the components to the data or the IoT devices is also registered allowing to demonstrate and show the access to the information of the devices.

5.5.5. Anomaly Detection

The Evrythng platform incorporates an intrusion detection system (IDS) which allows detecting common attacks, supported by Amazon Web Services (AWS) security features. This countermeasure entails the detection and response to attacks in real time.

In conclusion, Evrythng provides authentication functions to validate the identity of an IoT device, user and IoT components. Evrythng also provides mechanisms to protect data at move and rest. Authorization is also considered even if it includes just three operations. In the same way, Evrythng registers operations over data and IoT devices allowing accounting functionalities. Finally, Evrythng is able to detect anomalies through an IDS.

5.6. Dweet

5.6.1. Authentication

Regarding authentication from IoT devices to the cloud, Dweet supports connections from a wide set of devices included in a library of Dweet connectors. IoT devices need a lock (token) to connect to the platform. Locks can be used by more than one IoT device as long as they are not used simultaneously. Locks do not expire and they generate a monthly charge.

As for authentication between the users and the cloud, Dweet allows creating different users for whom it is mandatory to generate a password of at least six characters. After authentication, users can have access to the dashboard. Different operations over a user account can be done through the Dweet API such as add, edit or delete user accounts.

Finally, after a comprehensive review of the platform, we did not find information available regarding authentication between cloud components.

5.6.2. Encryption

With encryption of data at move, Dweet supports connections from the IoT device using the supported protocols HTTPS, HTTP or web sockets. HTTP does not implement encryption for the data, enabling a possible access to critical information by an attacker performing traffic sniffing.

In the case of encryption of data at rest, Dweet stores the reported values from the IoT devices in an unlimited storage.

5.6.3. Authorization

When dealing with authorization of operations over the IoT platform, users in Dweet can have three different roles: admin, manager and viewer. Each role has different privileges. The admin role can start operations to create, modify and delete IoT devices, device collections, alerts and users. This role can also buy locks and access the console. The manager role can start operations to create and modify IoT devices, device collections, and can access the console. Finally, the Viewer role can view IoT devices and collections as well as access the console.

After a comprehensive review of the platform, we did not find information available about the authorization of operations between components of the IoT platform.

5.6.4. Accounting

After a comprehensive review of the platform, we did not find information available, regarding either Accounting of operations from the users to the data or the IoT devices, or Accounting of operations from the components to the data or the IoT devices.

5.6.5. Anomaly Detection

Dweet allows defining alerts regarding the values being sensed to detect anomalies or unexpected behaviors. These alerts can use different means to inform the user such as SMS, email and text to speech.

In summary, Dweet supports authentication of IoT devices and users with the IoT platform but it does not support authentication between cloud components. Additionally Dweet is able to protect data at move even if the use of unsecured protocols is allowed. Regarding data at rest, Dweet does not indicate the use of cyphering. It also provides a structure of roles (admin, manager and viewer) which allows controlling some operations over the IoT components of the platform. Accounting is not implemented apparently. Finally, anomaly detection is implemented through alerts that allow monitoring IoT devices.

5.7. Node-Red

5.7.1. Authentication

As for authentication from IoT devices to the cloud, the platform asks for a different credential (either an API KEY or a serial) depending on the node type.

Authentication between the users and the cloud is considered because the user authentication with the Node-Red is done using a username and password which generate a token used for access. Passwords are stored within a configuration file as bcrypt hashes. Authentication can be also supported through an OAuth/OpenID provider such as Twitter or GitHub.

Regarding authentication between cloud components nodes and static content inside, Node-Red can be secured using basic authentication. In the configuration file, a username and password are defined and used to validate the access to the nodes.

5.7.2. Encryption

About encryption of data at move, information transferred through Node-Red can be ciphered using encryptor/decryptor nodes. These nodes can use end to end encryption with the AES-256 algorithm and a password provided by the user. Encryption can also be implemented in a workflow of Node-Red using a cryptographic library.

With regards encryption of data at rest, Node-Red provides storage API which allows storing information such as flow configuration, flow credentials, user settings, user sessions and node library content. Data could be stored in a cipher way using a cryptographic library.

5.7.3. Authorization

To achieve authorization of operations over the IoT platform, the permissions for each user authorized to connect to Node-Red can be set in the configuration file. There are different permission levels (e.g., read or write) which are based on a specific resource (e.g., flow or node).

On the other hand, to achieve authorization of operations between components of the IoT platform, nodes and static content can be secured through an username and password, which allows other Node-Red components to have a access to them.

5.7.4. Accounting

Accounting of operations from the users to the data or the IoT devices in Node-Red is supported thanks to a console logger which can be set through the configuration file. The logger requires the configuration of the following properties: level, metric and audit. Level refers to the six different logging levels that will be stored (fatal, error, warn, info, debug and trace). Metric allows loggin memory usage and received/sent events for each node. Finally, audit allows deciding whether API access events are logged including information like endpoint accessed, IP address, time stamp and requesting user.

As for accounting of operations from the components to the data or the IoT devices, the logging functions explained previously can be used when necessary to monitor the operations from other components of Node-Red over a node.

5.7.5. Anomaly Detection

Node-Red can make some basic anomaly detection using rules designed to monitor the values measured by the IoT device. After an anomaly has been detected, a message can be sent through an e-mail node. Additionally, it is possible to integrate Node-Red and BlueMix to implement an anomaly detection system. Anomaly detection is based on statistics considering mean and standard deviation of the values measured by the IoT device.

In conclusion, Node-Red provides authentication functions to validate the identity of users and cloud components. Node-Red also provides mechanisms to protect data at move and rest through the use of encryptor/decryptor nodes and cryptographic library. Authorization of users over the IoT platform and between components is also considered. Regarding accounting, Node-Red holds a console logger used to register operations to the data and the IoT devices. Finally, Node-Red is able to detect anomalies in the measured values through rules as well as an anomaly detection system using BlueMix.

5.8. Nimbits

5.8.1. Authentication

Regarding authentication from IoT devices to the cloud, IoT devices can send values to Nimbits using a POST request with an authentication header containing credentials used to authenticate the device.

As for authentication between the users and the cloud, users are created through a POST request to the Nimbits API containing the email address and password for the new users. Nimbits also supports user authentication through Google Accounts.

Finally, with regard to authentication between cloud components, the connection between the Nimbits system and its local instance of mySQL database is authenticated. New instances intending to connect to the database require running an authentication routine where the IP address of the connecting Nimbits instance is set, in addition to the SQL user/password set in the Nimbits configuration file.

5.8.2. Encryption

By default, Nimbits uses HTTP which does not offer protection for the data at move. HTTP is employed for the communication between the users and the platform. Posting of snapshot (set of attributes), e.g., GPS coordinates or a time stamp, is also done through HTTP.

On the other hand, regarding encryption of data at rest, by default Nimbits uses mySQL database allowing to set it according to the security needs. Even if it is not explicitly mentioned, mySQL database could be set to store data in a cypher way. Additionally, users passwords are stored with a forward only encryption and cannot be recovered, only reset.

5.8.3. Authorization

When dealing with authorization of operations over the IoT platform, Nimbits uses two roles: administrator and regular users. The only difference between these two roles is that regular users cannot make CRUD operations over other users. Both roles can call delete, create and update methods in the REST API and use the methods from a Nimbits client. Users are allowed to see IoT data associated to another user after an invitation/approval process.

As for authorization of operations between components of the IoT platform, Nimbits allows having a load balanced environment composed by many instances of Nimbits. The access of these instances over the work is managed with a configuration file that enables each instance and controls processing of delay time, alert access rate and simultaneous alert processing rate. Such configuration file also allows configuring the database connections supporting Nimbits.

5.8.4. Accounting

To achieve accounting of operations from the users to the data or the IoT devices, Nimbits uses Slf4j (Simple Logging Facade for Java) to log operations. Slf4j provides a Java logging API that is consumed by Nimbits and permits five logging levels (ERROR, WARN, INFO, DEBUG, and TRACE). Nimbits documentation does not explain whether this logging is intended to account operations from users over the data or the IoT devices.

After a comprehensive review of the platform, we did not find information available regarding the Accounting of operations from the components to the data or the IoT devices.

5.8.5. Anomaly Detection

Nimbits is by default a rule engine which allows monitoring values reported by IoT devices. Nimbits can evaluate rules which can trigger events under specific conditions. An alert can be generated when new values are not received during an idle alert time that is defined as a value in seconds. Rules can be set to generate email alerts, push notification, statistics or any calculation.

Nimbits supports the authentication between IoT devices, users and cloud using credentials. However, data are transported through HTTP which does not offer protection against confidentiality. Data at rest could be secured through a database protection even if it is not mentioned in the documentation of Nimbits. It also allows two different user roles and a scheme to manage Nimbits instances. Regarding accounting, Nimbits logs the operations initiated by the user over the data or the IoT devices, although information regarding logging of operations between component was not found. Finally, Nimbits contains a rule engine to monitor changes in the values reported by the IoT devices.

5.9. The Thing System

5.9.1. Authentication

To achieve authentication from IoT devices to the cloud, devices are authenticated using a universally unique identifier (UUID) which is defined in the device creation process. The creation process can be done by an authorized client sending a request containing the UUID, the device name, device parameters and the taxonomy which is used to identify the device type.

As for authentication between the users and the cloud, clients must be authenticated with the cloud to consume API rest services. In the context of The Thing System, a client can have many associated users and can be authenticated using a One Time Password (OTP). Thus, applications such as Google Authenticator running from smart phones can be used to secure the authentication process.

Finally, after a comprehensive review of the platform, we did not find information available about the authentication between cloud components.

5.9.2. Encryption

Regarding encryption of data at move, the communication between client and steward (cloud site server) is done through HTTPS. Clients are able to retrieve a "privacy package" from steward containing: SNI (Server Name Indication) hostname, self-signed public-key certificate and an SSH key fingerprint. Likewise, clients can upload a SSH public key fingerprint to the steward. After HTTPS is negotiated, the client upgrades to WebSockets and then is able to consume API REST services provided by The Thing System. Communication between devices and The Thing System is done using a protocol called Simple Thing Protocol transported through web sockets implementing an encryption protocol that protects the exchanged information.

After a comprehensive review of the platform, we did not find information available with regard to encryption of data at rest.

5.9.3. Authorization

To deal with authorization of operations over the IoT platform, the Thing System supports user authorization through a mechanism of roles. A user can have one of five roles: master (unlimited access to the steward), resident (extended access to the steward), guest (limited access to the steward), device (devices implementing the Simple Thing Protocol) and cloud (for services in the cloud).

After a comprehensive review of the platform, we did not find information available regarding authorization of operations between components of the IoT platform.

5.9.4. Accounting

To achieve accounting of operations from the users to the data or the IoT devices, the developer console is the module in The Thing System used to see the log information regarding the IoT devices. In the developer console, it is possible to see all the available devices and the reports from them.

On the other hand, to achieve accounting of operations from the components to the data or the IoT devices, logging of operations is done with a syslog style which allows managing different functions (alert, critical, error, warning, notice, informational, and debug). The log can register two arguments: a string and a property-list object. The log operations are set inside the module file which contains the definition of an object for each created device.

5.9.5. Anomaly Detection

The steward has a rule engine able to observe the values and events reported by the devices and perform specifically scheduled tasks according to the detected values. Rules are evaluated by apprentices that are intelligent agents evaluating current conditions and performing actions.

The Thing System includes authentication between IoT devices and the cloud using an UUID. Authentication between users and the cloud can be done through an OTP. Communication between client and the cloud is done by HTTPS and WebSocket, and communication between the IoT devices and the cloud is done using Simple Thing Protocol which implements encryption. The Thing System also includes a mechanism of roles over which the authorization is based. Information regarding authorization of operations between components of the IoT platform was not found. Accounting is also supported by The Thing System through the developer console which allows accessing the log information generated by the devices. This platform also contains a rule engine which monitors values reported by the devices allowing an anomaly detection.

5.10. Sitewhere

5.10.1. Authentication

To achieve authentication from IoT devices to the cloud, IoT devices can be registered with Sitewhere through JSON structure data which contain a hardware ID that is a unique identifier for the device, a token that specific the type of device being registered and a token that specifies the site where the device will be associated. When a device registers with Sitewhere, it sends back a response to the device confirming the registration. Thus, the identifier and tokens are used to authenticate the device.

Regarding authentication between the users and the cloud, users require a username and password to be authenticated with Sitewhere and to be able to connect to the administrative console and to access the REST services. Additionally, a token is also needed to validate that the user is authorized to call a service from the API REST provided by Sitewhere.

After a comprehensive review of the platform, we did not find information available about authentication between cloud components.

5.10.2. Encryption

To provide encryption of data at move, communication between devices and Sitewhere is done in uncipher communication. The communication can be developed through different protocols like HTTP, WebSockets, direct sockets, MQTT and AMQP (Advanced Message Queuing Protocol).

After a comprehensive review of the platform, we did not find information available with regard to Encryption of Data at Rest.

5.10.3. Authorization

As for authorization of operations over the IoT platform. permissions are assigned to Sitewhere users, also called granted authorities, which allow access to different functionalities. When a user is created, a list of permissions can be selected. User requires a tenant authorization token to access functionalities in Sitewhere. Authorization is implemented with the Java EE Framework Spring security.

After a comprehensive review of the platform, we did not find information available dealing with authorization of operations between components of the IoT platform.

5.10.4. Accounting

Regarding accounting of operations from the users to the data or the IoT devices, when a user starts an operation over Sitewhere entities, the username of the authenticated user is stored to keep track of all his actions.

After a comprehensive review of the platform, we did not find information available regarding accounting of operations from the components to the data or the IoT devices.

5.10.5. Anomaly Detection

Sitewhere has a zone-test-event-processor which is used to validate if location events are inside a predefined zone. The anomaly of location can be detected through a previously defined zone-test element that includes: (i) a zoneToken which is unique for each zone; (ii) a condition to test which can be inside or outside; (iii) an alert type; (iv) an alert level; and (v) an alert message.

Sitewhere performs authentication from IoT devices to the cloud using tokens. Authentication between users and the cloud is done through credentials which also allow the users to get a token to perform calls to REST services. Encryption of data at move and rest is apparently not done. Authorization is developed through granted authorities which are required to access to Sitewhere functionalities. Authorization of operations between components of the IoT platform apparently is not done. Sitewhere keeps track of user operations, even if there is no information about the accounting done to operations started by the components of the cloud. Finally, Sitewhere allows detecting anomalies related to the location of the IoT device.

5.11. Summary

Next, a summary comparison of the above solutions is presented. Additionally, Table 1 offers a summary of this comparison and can be used as a quick reference.

For the first security criterion, authentication, we can conclude that all analyzed platforms provide mechanisms to authenticate the IoT devices and the users with the cloud, while the authentication between cloud components is only present in some platforms: Samsung Artik, IBM Watson, Oracle IoT, Evrythng, Node-Red and Nimbits. The most common mechanisms selected to authenticate those IoT devices are OAuth2, credentials, API keys, tokens, authentication header, UUID and hardware ID. In turn, the most common mechanisms selected to authenticate the end users are credentials, certificates and OAuth tokens. Finally, authentication between components is done mainly through tokens, LDAP, PKI and credentials.

Regarding the second security criterion, encrypted information management, all the platforms, with exception of Nimbits and Sitewhere, have some encryption mechanism for data at move, being the most frequently ones TLS or HTTPS. Some platforms include other mechanisms such as SSL (Samsung Artic and IBM Watson), IPSec (IBM Watson), symmetric cipher algorithms including AES (IBM Watson, Everythng and Node-Red), asymmetric cipher algorithms including RSA (IBM Watson and Oracle IoT), secure WebSockets (Samsung Artik and Dweet), Tokens (Oracle IoT) and the Simple Thing Protocol (The Thing System). On the other hand, regarding encryption in data at rest, most of the platforms implement symmetric and asymmetric algorithms, typically AES and RSA. However, three of the reviewed platforms (Dweet, The Thing System and Sitewhere) did not provide any information with respect to encrypted information management.

As for the third security criterion, authorization, all platforms implement some kind of authorization scheme, discretionary or role based, for the control of operations over the IoT platform. These schemes are based mainly on policies, rules or lists and, depending on the mechanism implemented by each platform, it can eventually be possible to have fine-grained directives that control the specific user actions over the general IoT ecosystem. Additionally, there is a big diversity of mechanisms to control the authorization between components of the IoT platform, having each platform a different one. However, it is important to mention that some platforms (Amazon IoT, Dweet, The Thing System and Sitewhere) do not have information regarding this authorization between components.

Concerning the fourth security criterion, accounting, eight of ten examined platforms use logging solutions to maintain information about the user activity to the data or the IoT devices. The difference between each one is related to the levels of detail that is stored in the logs and the use of a third-party solution to analyze such traces and take actions accordingly. Oracle IoT and Dweet are the only two surveyed platforms that do not indicate any information about this matter. Furthermore, regarding the accounting of component activities to the data or the IoT devices, five of ten platforms (IBM Watson IoT, Oracle IoT, Evrythng, NodeRed and The Thing System) use log solutions, being Oracle IoT the most outstanding platform implementing analytics over the data gathered in the logs. The remaining platforms (Samsung Artik, Amazon IoT, Dweet, Nimbits and Sitewhere) do not indicate any information about the accounting of activities coming from components of the IoT ecosystem.

Finally, regarding the fifth security criterion, anomaly detection, all platforms except Oracle IoT use some kind of mechanism to identify unusual behavior on the platform. In all cases, these solutions are based on the monitoring of the operations of the platform to identify anomalies (e.g., change of measured values, inputs coming from external components, connection or disconnection of components, etc.). Some platforms even offer a specific mechanism focused on cybersecurity events, such as IBM Watson which is able to scan the asset environment for vulnerabilities, or Evrythng which integrates an intrusion detection system. Moreover, some platforms can also offer mechanisms to alert the platform administrator about an anomaly, e.g., using SMS or emails. It is worth mentioning that the anomaly detection feature can be provided by the IoT platform itself, or it can be offered through an independent solution that is integrated with the IoT platform.

In conclusion, it is observed that the platforms that have the highest compliance with the security criteria evaluated are IBM Watson, Evrythng and Node-Red, fulfilling all the selected security criteria, followed by Samsung Artik, Amazon IoT and Oracle IoT, among others. Besides, it is also possible to affirm that there is no direct relation between the compliance with a security criteria and the recognition of the manufacturer, existing some relatively new competitors offering platforms with the same or better security level than the traditional ones. It is also important to note that the documentation available for the platforms is an important aspect to consider when a new IoT service must be built, because this is critical to support the proper analysis, design, implementation and testing of safe IoT services.

Table 1. Platforms evaluation results.

Platform	Authentication System	Encrypted Information	Authorization System	Accounting System	Anomaly Detection
Samsung Artik [36]	Devices IoT to cloud: OAuth2	Data at move: DTLS, socket	Operations over IoT platform: tokens	Users to the data or the IoT devices: record data timestamp	Change Detection: Anodot
	Users and the cloud: client credentials	Data at rest: encryption keys	Operations between components of the IoT platform: cloud apps	Components to the data or the IoT devices: N/A	
	Cloud components: tokens to execute API calls				
Amazon IoT [38]	Devices IoT to cloud: device credentials	Data at move: TLS	Operations over IoT platform: polices	Users to the data or IoT devices: CloudWatch, CloudTrail	Change Detection: AWS Lambda
	Users and the cloud: user credentials	Data at rest: AES-256, digital certificates	Operations between components of the IoT platform: N/A	Components to the data or the IoT devices: N/A	
	Cloud components: N/A				
IBM Watson IoT [40]	Devices IoT to cloud: API keys	Data at move: MQTT and TLS, SSL or IPSec, AES, RSA	Operations over IoT platform: polices, blacklists and white-lists	Users to the data or the IoT devices: logs	Change Detection: Nessus
	Users and the cloud: ID/password, OAuth tokens and certificates	Data at rest: SSL certificates and proprietary solutions	Operations between components of the IoT platform: spaces for authenticated clients	Components to the data or the IoT devices: logs	
	Cloud components: LDAP, IBM web identity				
Oracle IoT [42]	Devices IoT to cloud: OAuth	Data at move: HTTPS and JSON Web Token, RSA	Operations over IoT platform: roles and policies	Users to the data or the IoT devices: N/A	Change Detection: N/A
	Users and the cloud: credentials	Data at rest: Trusted Assets Manager, cipher algorithms	Operations between components of the IoT platform: OAuth2 access token	Components to the data or the IoT devices: activity logs	
	Cloud components: Certificates, OAuth tokens				
Evrythng [45]	Devices IoT to cloud: API key	Data at move: TLS and AES	Operations over IoT platform: access, manage and sharing rules	Users to the data or the IoT devices: logs	Change Detection: IDS
	Users and the cloud: 2FA, TOTP and social media authentication.	Data at rest: key management infrastructure	Operations between components of the IoT platform: manage and sharing rules	Components to the data or the IoT devices: logs	
	Cloud components: Public Key Infrastructure				

Table 1. *Cont.*

Platform	Authentication System	Encrypted Information	Authorization System	Accounting System	Anomaly Detection
Dweet [46]	Devices IoT to cloud: tokens	Data at move: HTTPS and web sockets	Operations over IoT platform: roles	Users to the data or the IoT devices: N/A	Change Detection: SMS or e-mail notification
	Users and the cloud: ID/password	Data at rest: N/A	Operations between components of the IoT platform: N/A	Components to the data or the IoT devices: N/A	
	Cloud components: N/A				
Node-Red [48]	Devices IoT to cloud: API keys or serials	Data at move: AES-256	Operations over IoT platform: user permissions	Users to the data or the IoT devices: logs	Change Detection: anomaly detection, Bluemix
	Users and the cloud: username and password	Data at rest: cryptographic libraries	Operations between components of the IoT platform: username and password	Components to the data or the IoT devices: logs	
	Cloud components: basic authentication				
Nimbits [50]	Devices IoT to cloud: Authentication Header	Data at move: NO	Operations over IoT platform: roles	Users to the data or the IoT devices: Slf4j log schema	Change Detection: detection rules
	Users and the cloud: email and password, or Google Accounts	Data at rest: cyphering mySQL database	Operations between components of the IoT platform: permission for load balancing	Components to the data or the IoT devices: N/A	
	Cloud components: SQL user/password				
The Thing System [52]	Devices IoT to cloud: UUID	Data at move: HTTPS and Simple Thing Protocol	Operations over IoT platform: roles	Users to the data or the IoT devices: logs	Change Detection: detection rules
	Users and the cloud: username, password and OTP Cloud components: N/A	Data at rest: N/A	Operations between components of the IoT platform: N/A	Components to the data or the IoT devices: syslog	
Sitewhere [54]	Devices IoT to cloud: hardware ID and tokens.	Data at move: N/A	Operations over IoT platform: granted authorities	Users to the data or IoT devices: tracking user operations	Change Detection: zoneToken
	Users and the cloud: username and password	Data at rest: N/A	Operations between components of the IoT platform: N/A	Components to the data or the IoT devices: N/A	
	Cloud components: N/A				

6. Description of the IoT Service to Implement

This Section describes an IoT service which can be implemented in two of the IoT platforms reviewed previously which have outstanding security features. The service to implement is based in the regrettable event on 28 November 2016, near the city of Medellin (Colombia), where the aerial tragedy of Chapecoense happened (http://www.bbc.com/news/world-latin-america-38142998). In that event, 76 people died, including several players of the Brazilian football team "Chapecoense Football Association" due to an airplane accident, which was apparently caused when the aircraft had a very low level of fuel and the pilot did not make the report immediately. If he had reported the situation promptly, an emergency protocol would have been launched and the aircraft would have had priority to land.

Based on this incident and motivated to avoid similar situations, a Commercial Flight Monitoring Service is proposed using an IoT platform and helping to guarantee the flight safety. This service must allow monitoring different aspects of the aircraft through different sensors allowing the generation of early warnings and managing some actuators installed on the aircraft to control some variables automatically. Sensors and actuators are currently installed in the aircraft through avionic devices. Avionics [56] is an abbreviation for the term electronics for aviation and comprises electronic systems for communication, navigation, information display and lighting, among many other systems distributed throughout the aircraft, and estimated to be 30% of the total cost of the aircraft. Avionic devices with connection capabilities could be considered IoT devices which can compose an IoT service. This IoT service assumes that the aircraft has the following IoT devices:

1. Collision management device: It measures the presence of objects within a horizontal and vertical space.
2. Cabin pressure device: It measures the pressure level in the cabin.
3. Altitude management device: It measures the altitude at which the airplane is located.
4. Fuel level management device: It measures the fuel level of the aircraft.
5. Engine temperature management device: It measures the temperature of the engine.
6. Engine coolant management device: It measures the coolant level of the engine.
7. Storm detection device: It has two sensors: (i) a distance sensor which calculates the distance from the aircraft to a storm; and (ii) a luminosity sensor which allows determining the severity of the storm based on its luminosity.
8. Fire detection device: It is a heat sensor that determines whether there is presence of fire in the aircraft.
9. Navigation system: This device has geolocation functions, and reports the aircraft latitude and longitude.

An IoT service is based on rules to control the variables being measured in the aircraft. The Commercial Flight Monitoring Service defined here is based on the following rules:

1. Rule 1: An alarm must be generated when the aircraft finds objects within a horizontal space less than 9.25 km and within a vertical space less than 305 m.
2. Rule 2: The airplane must auto regulate the pressure in the cabin to keep it within the range [75.3–81.2] kilo Pascal.
3. Rule 3: An alarm must be generated when the aircraft is below 4 km of altitude and is at a distance from the destination or origin greater than 120 km.
4. Rule 4: An alarm must be generated and sent to the nearest control tower when the aircraft is at a fuel level below 20% of the tank capacity and the distance to arrival destination is bigger than 1000 km.
5. Rule 5: An alarm must be generated when the engine temperature is bigger than 200 degrees Celsius.
6. Rule 6: The airplane must auto regulate the level of the engine coolant to keep it above 70%. When the liquid level is under that level, an alarm must be generated.

7. Rule 7: The aircraft must generate an alarm when there is a storm at less than 50 km with a severity lower than 2 on a scale of 0–5, where 0 means no severe thunderstorm and 5 means widespread severe storms, according to the scale from the Storm Prediction Center (https://www.spc.noaa.gov/).
8. Rule 8: The aircraft must detect and report the presence of fire in the engine compartment. In the case of fire, the retardant foam must be activated.
9. Rule 9: The plane must constantly report its geographical position.

Figure 1 describes the components of the Commercial Flight Monitoring service.

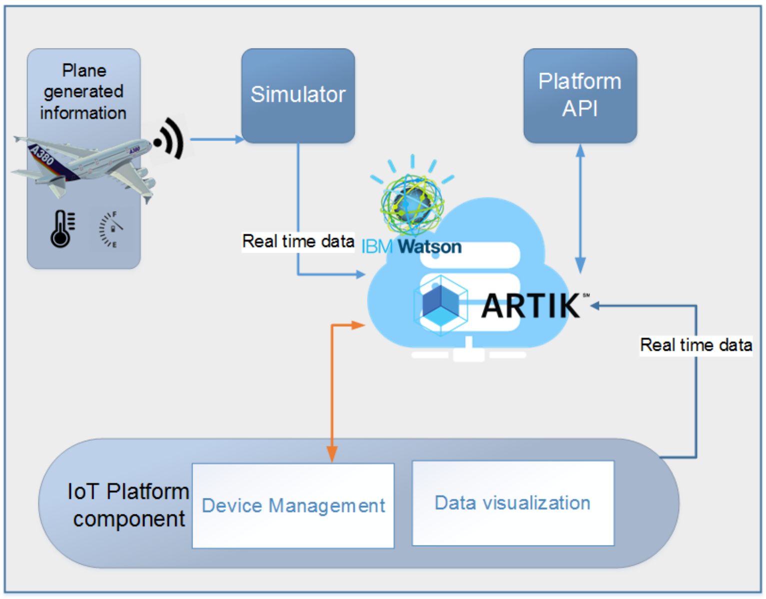


Figure 1. Topology corresponding to the scenario implemented.

The IoT service is aimed to have the following user roles: aeronautical agent, air traffic controller and emergency pilot. An aeronautical agent must be able to read all the information reported by the IoT devices. An air traffic controller must be able to read altitude management device and Navigation system. Finally, an emergency pilot must be able to read from all the IoT devices and send commands to these devices.

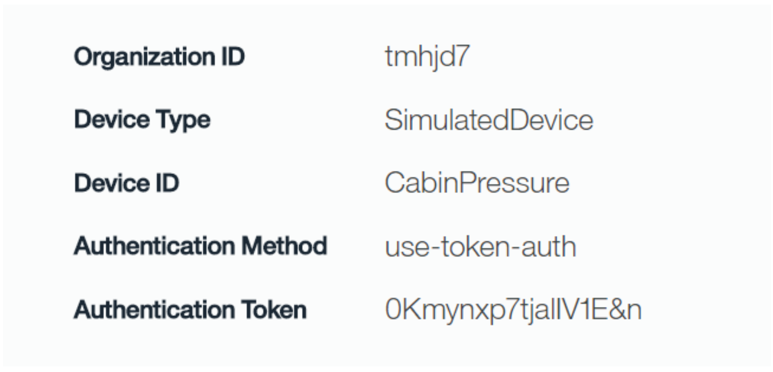
7. Implementation of the Service

The process of implementing the IoT service described in Section 6 on two IoT platforms, Samsung Artik and IBM Watson IoT, is illustrated below. These platforms are between the ones that met the most security criteria in the evaluation carried out in Section 5, having just one criteria marked as N/A for the case of Samsung Artik. Another platforms also fit the most of the security criteria (Evrythng and Node-Red), however Samsung Artik and IBM Watson IoT offered more documentation and support for the implementation. Additionally, is important to notice that IBM Watson IoT and Samsung Artik offer security features that are essential for a Commercial Flight Monitoring Service: (i) authentication between IoT devices, users and cloud components through strong mechanism; (ii) encryption of data at move and at rest using resilient cypher algorithms; (iii) authorization of actions over the IoT platform using different proven techniques; (iv) accounting of activities from the user toward the IoT devices through rich logs; and (v) dedicated modules for anomaly detection purposes.

7.1. IBM Watson IoT

The setting of the IoT service is done using the web interface of IBM Watson IoT (https://console.bluemix.net/).

Authentication between the IoT devices and the cloud requires first registering the device in the cloud to obtain some essential information (organization ID, Device Type, Device ID, Authentication Method and Authentication Token) which is used later to setup the IoT device. Authentication token can be auto-generated or self-provided. An example of an IoT device registered in the cloud is shown in Figure 2.



Organization ID	tmhjd7
Device Type	SimulatedDevice
Device ID	CabinPressure
Authentication Method	use-token-auth
Authentication Token	0Kmynxp7tjallV1E&n

Figure 2. Cabin pressure IoT device registration information.

For simplicity, IoT devices are being simulated using a node.js application that connects and sends data to IBM Watson IoT. The environment file of this Node.js application must be set according to the essential information obtained previously and represented in Figure 2. After a proper configuration of the IoT device, it becomes registered and it starts sending measured values to the IoT service, as shown in Figure 3.

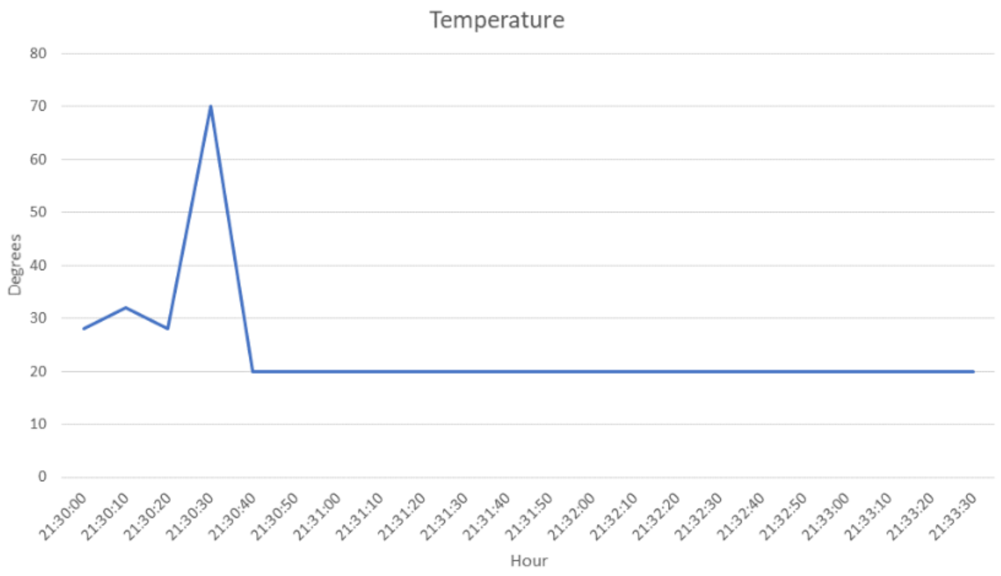
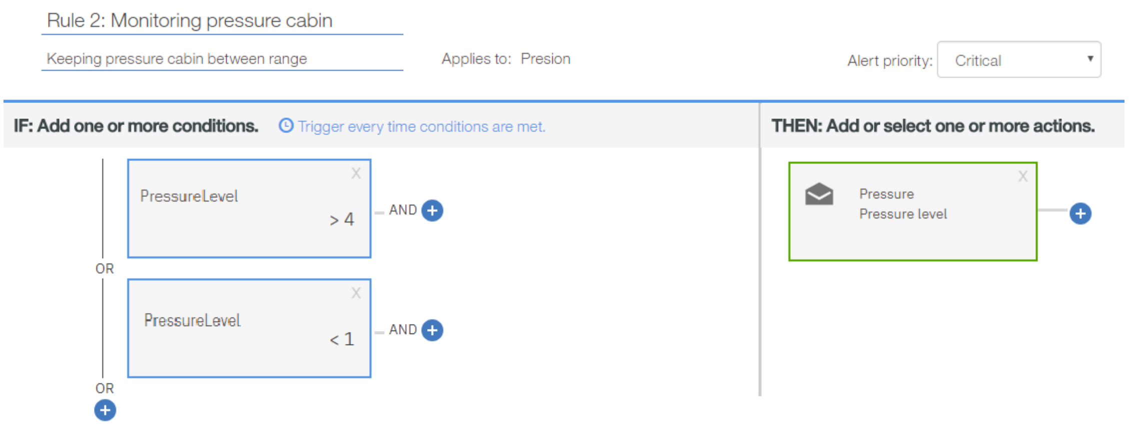


Figure 3. Registered IoT device sending data to the cloud.

Once all the IoT devices are registered, the service rules must be created. A rule in IBM Watson IoT is composed of one or more conditions that must be fulfilled to trigger the rule, and one or more actions that are executed in case the conditions are satisfied. For example, Rule 2, associated to the cabin pressure device, states that the airplane must auto regulate the pressure in the cabin to keep it within the range 1–4. Thus, it requires setting one condition that monitors pressure values within the range, and in case the condition is not met, set an action to produce an alarm and regulate the cabin pressure. Rule 2 can be seen in Figure 4.


Rule 2: Monitoring pressure cabin

Keeping pressure cabin between range

Applies to: Presion

Alert priority: Critical

IF: Add one or more conditions. Trigger every time conditions are met.

PressureLevel > 4 AND

OR

PressureLevel < 1 AND

OR

THEN: Add or select one or more actions.

Pressure

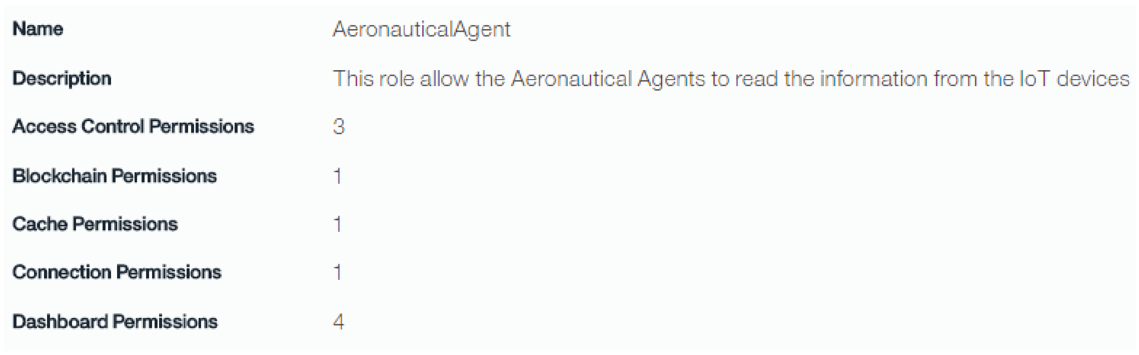
Pressure level

Figure 4. Rule implemented to monitor pressure cabin.

IBM Watson IoT supports roles which allow defining different aspects around the IoT Service management, including the permissions over the IoT devices, and which can be referenced in the creation of new users. In the context of our service, it was required to create the role of "aeronautical agent" which can read all the information reported by the IoT devices. The "aeronautical agent" role was set to be able to view device properties, view live data from cache, view the client connection state, subscribe to events from a device from an external platform, create/update/view dashboards and view historical data, and view actions, alerts, schemas, rules and devices. Another roles can be created as required by the service.

Aeronautical agent role can be seen in Figure 5, which specifies the number of permission defined for such role for each permission category. The permission categories are following:

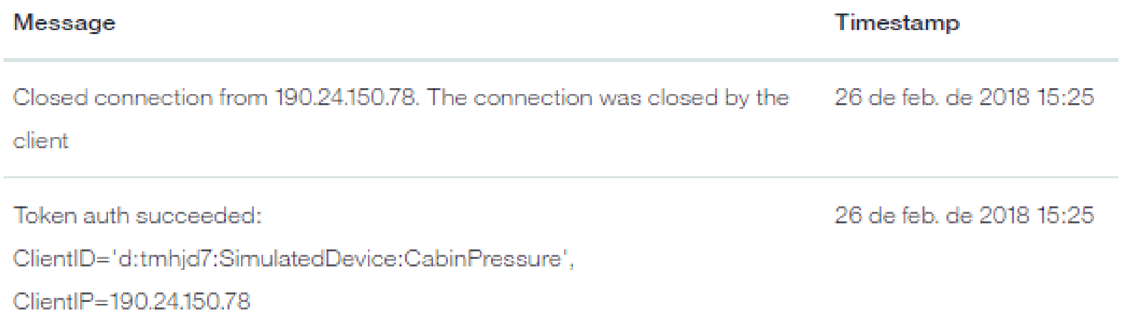
- Access Control Permissions: It defines permissions to create, update and delete different features in the applications such as passwords, devices, members and roles.
- Blockchain Permissions: It defines permissions to manage the blockchain configuration.
- Cache Permissions: It defines permissions to manage the active data (cache).
- Connection Permissions: It defines permissions to manage the client connection status.
- Dashboard Permissions: It defines permission to create, delete, share, update and see dashboards.



Name	AeronauticalAgent
Description	This role allow the Aeronautical Agents to read the information from the IoT devices
Access Control Permissions	3
Blockchain Permissions	1
Cache Permissions	1
Connection Permissions	1
Dashboard Permissions	4

Figure 5. Aeronautical agent role definition.

IBM Watson IoT has accounting functions to monitor the activities that are done over the IoT devices, which are stored in log files. Different roles have certain accounting capabilities: viewing server logs, viewing diagnostic logs and viewing operations. For each device, there is a connection log which allows viewing the activity between the device and the cloud. Figure 6 shows the connection log for the cabin pressure device.



Message	Timestamp
Closed connection from 190.24.150.78. The connection was closed by the client	26 de feb. de 2018 15:25
Token auth succeeded: ClientID='d:tmhjd7:SimulatedDevice:CabinPressure', ClientIP=190.24.150.78	26 de feb. de 2018 15:25

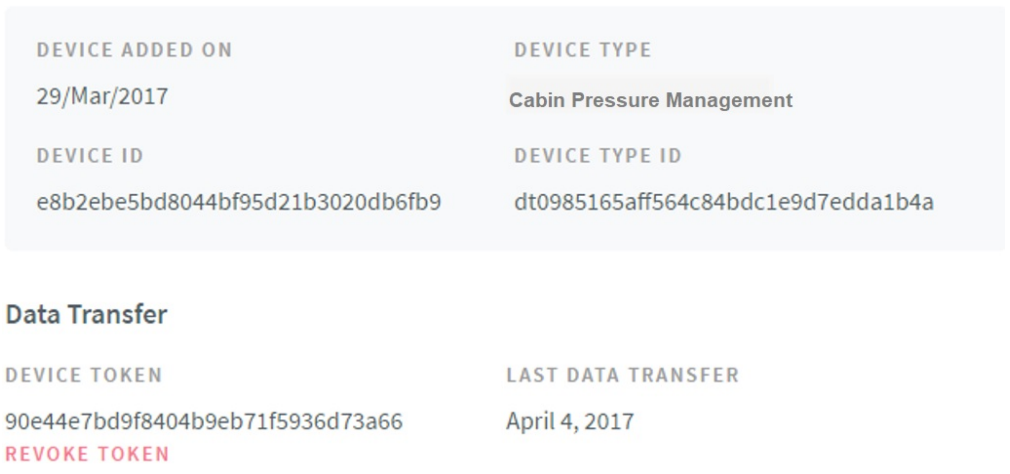
Figure 6. Connection log from Cabin Pressure device.

IBM Watson IoT enables a secure implementation of the Commercial Flight Monitoring Service described in Section 6, mainly because it offers features to fit each one of the security criteria defined previously. Authentication is achieved through the use of authentication tokens (applicable to the IoT devices installed in the aircraft) and user accounts (applicable to end users and components connecting to the Commercial Flight Monitoring Service). Encryption is achieved through different protocols such as MQTT with TLS, PPTP, SSL or IPSec (for data exchange between avionic devices and IBM Watson Cloud). Authorization is implemented through access control policies (used to define roles with allowed operations over the avionic devices). Accounting is implemented through the connection logs existing for each avionic device. Finally, even if not shown in this section, anomaly detection can be achieved through Nessus.

7.2. Samsung Artik Cloud

The Samsung IoT platform allows the creation, configuration and subsequent simulation of different IoT devices; in this case, the implementation of the sensors or data elements mentioned in the definition of the service carried out in Section 6.

Authentication between IoT devices and Samsung Artik Cloud requires first registering the device, generating the following information: Device ID, Device Type, Device Type ID and Device Token (auto-generated). This information is used to setup the IoT device. Device token is auto-generated. An example of an IoT device registered in Samsung Artik Cloud is shown in Figure 7.



DEVICE ADDED ON	DEVICE TYPE
29/Mar/2017	Cabin Pressure Management
DEVICE ID	DEVICE TYPE ID
e8b2ebe5bd8044bf95d21b3020db6fb9	dt0985165aff564c84bdc1e9d7edda1b4a

Data Transfer

DEVICE TOKEN	LAST DATA TRANSFER
90e44e7bd9f8404b9eb71f5936d73a66	April 4, 2017

Figure 7. Cabin pressure IoT device registration information.

For simplicity, IoT devices are being simulated using the Device Simulator included in Samsung Artik which is an application able to emulate data with a specific interval and following a selected data pattern (random, increment, constant value and cycle). To emulate a Cabin Pressure device, an interval of 5000 ms has been configured and a random pattern selected a value between 1 and 10. After a proper configuration of the IoT device, the simulation can start and the data will be received by the cloud, as shown in Figure 8.

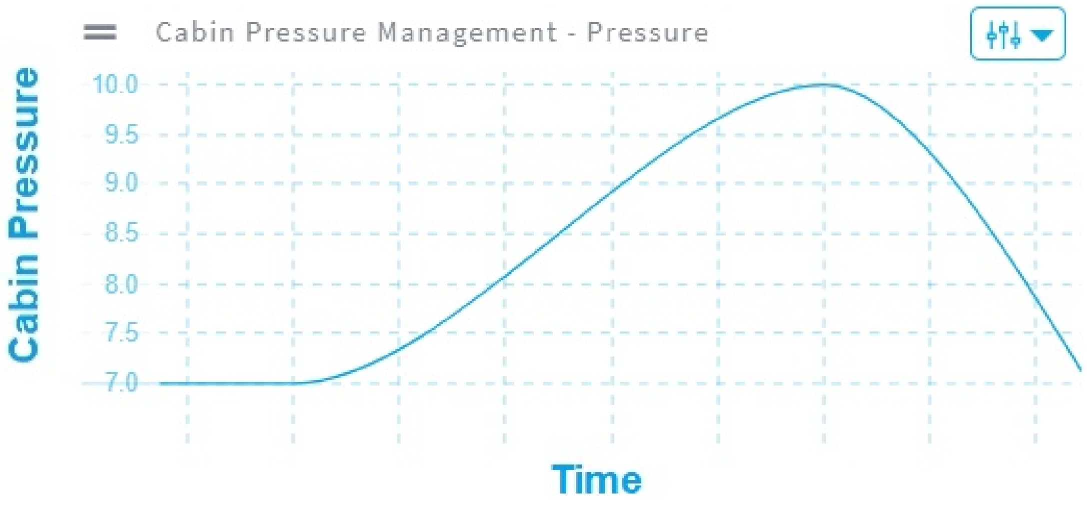
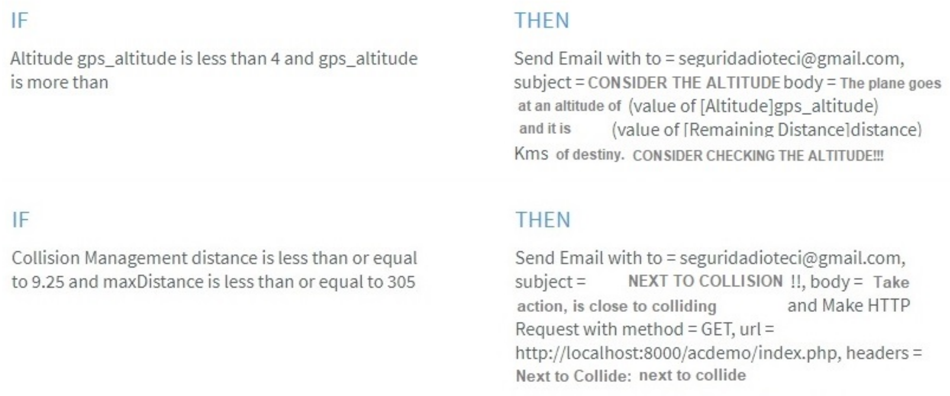


Figure 8. Registered IoT device sending data to Cloud.

Continuing the implementation, Samsung Artik Cloud allows defining a device operation rules. These rules allow defining an alarm in the case of an anomaly. Figure 9 shows some of the implemented rules for the avionics service, within the Samsung Artik platform.



IF

Altitude gps_altitude is less than 4 and gps_altitude is more than

THEN

Send Email with to = seguridadioteci@gmail.com, subject = CONSIDER THE ALTITUDE body = The plane goes at an altitude of (value of [Altitude]gps_altitude) and it is (value of [Remaining Distance]distance) Kms of destiny. CONSIDER CHECKING THE ALTITUDE!!!

IF

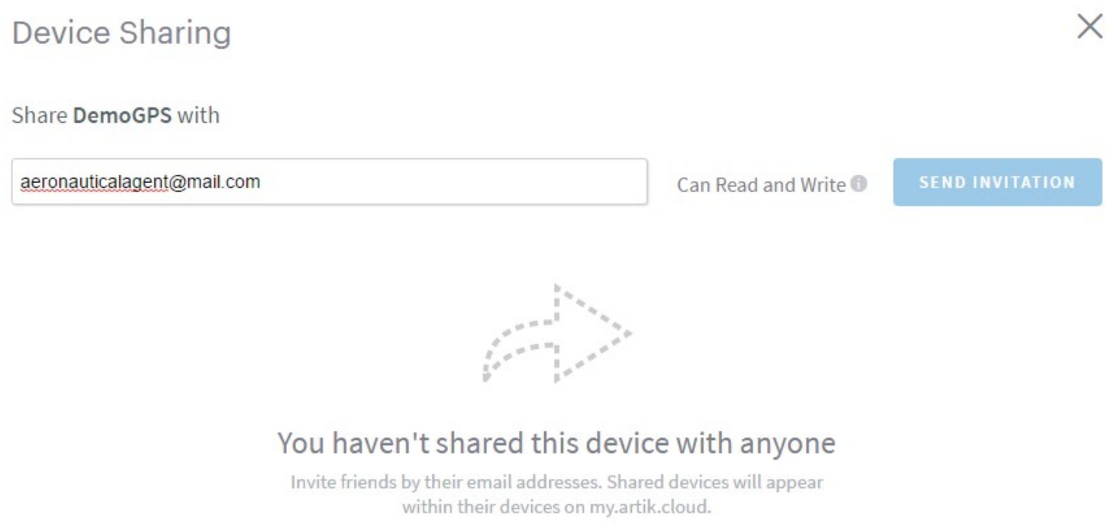
Collision Management distance is less than or equal to 9.25 and maxDistance is less than or equal to 305

THEN

Send Email with to = seguridadioteci@gmail.com, subject = NEXT TO COLLISION !!, body = Take action, is close to colliding and Make HTTP Request with method = GET, url = http://localhost:8000/acdemo/index.php, headers = Next to Collide: next to collide

Figure 9. Rules on Samsung Artik platform.

Samsung Artik allows defining with whom to share the information of the different IoT devices, which allows having different profiles of users or consumers for the service (Figure 10). Samsung Artik has an API that can be used to connect a web or mobile application, which can perform operations on data or IoT devices.



Device Sharing

Share **DemoGPS** with

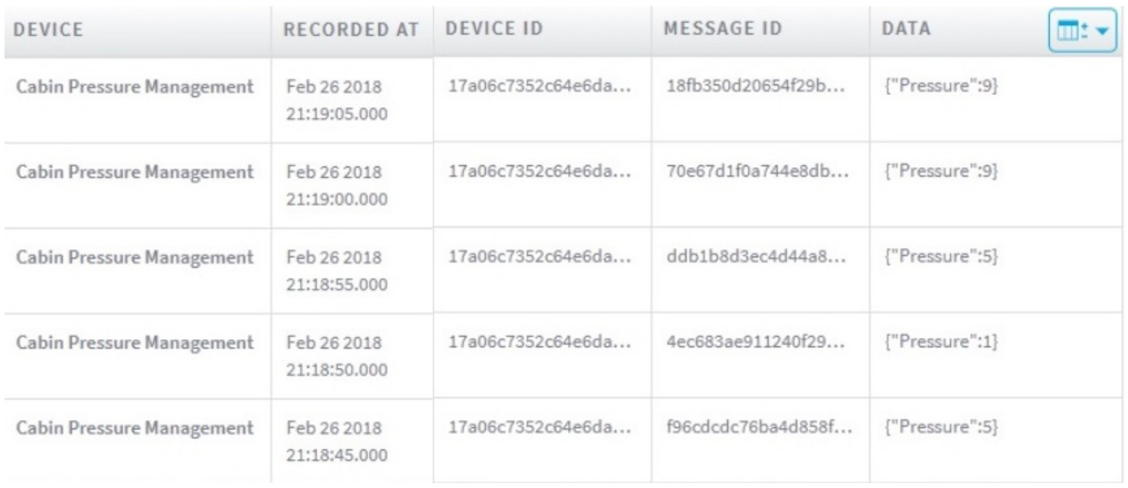
aeronauticalagent@mail.com

You haven't shared this device with anyone

Invite friends by their email addresses. Shared devices will appear within their devices on my.artik.cloud.

Figure 10. Sharing navigation system device with the aeronautical agent.

Samsung Artik has some accounting functions which allow reviewing the messages that have been received from the IoT devices and the actions that have been generated when a rule is matched. For each device, a set of logs can be generated containing Device, Device Type, Date and time when the message was recorded; Date and time when the message was received; Device ID; Message ID; and Data. Figure 11 shows the message and action logs for the cabin pressure device.



DEVICE	RECORDED AT	DEVICE ID	MESSAGE ID	DATA
Cabin Pressure Management	Feb 26 2018 21:19:05.000	17a06c7352c64e6da...	18fb350d20654f29b...	{"Pressure":9}
Cabin Pressure Management	Feb 26 2018 21:19:00.000	17a06c7352c64e6da...	70e67d1f0a744e8db...	{"Pressure":9}
Cabin Pressure Management	Feb 26 2018 21:18:55.000	17a06c7352c64e6da...	ddb1b8d3ec4d44a8...	{"Pressure":5}
Cabin Pressure Management	Feb 26 2018 21:18:50.000	17a06c7352c64e6da...	4ec683ae911240f29...	{"Pressure":1}
Cabin Pressure Management	Feb 26 2018 21:18:45.000	17a06c7352c64e6da...	f96cdcdc76ba4d858f...	{"Pressure":5}

Figure 11. Message and action logs for Cabin Pressure device.

Samsung Artik enables a secure implementation of the Commercial Flight Monitoring Service described in Section 6, mainly because it offers features to fit each of the security criteria defined previously. Authentication is achieved through the use of device tokens (applicable to the IoT devices installed in the aircraft), user accounts (applicable to end users connecting to the Commercial Flight Monitoring Service) and API keys (applicable for connection to the Samsung Artik API). Encryption is achieved through the use of DTLS and SSL (for data exchange between avionic devices and Samsung Cloud) and through a TrustZone (to guarantee a secure storage). Authorization is implemented through OAuth2 access tokens (used to define profiles with allowed operations over the avionic devices). Accounting is implemented though the logs existing for each avionic device. Finally, even if not shown in this section, anomaly detection can be achieved through Anodot.

8. Conclusions

The IoT paradigm of "everything connected together" is changing our lives. Several application domains have already been enhanced, and many enterprises foresee great economic benefit in the near future [57]. This belief is reflected in the market-place, where several players proposed their own IoT platform to develop various IoT services. Consequently, choosing the more suitable IoT platform for the implementation is not a trivial task for researchers and developers. Moreover, the security features exposed by these platforms represent a crucial parameter to select the most appropriate solution. To this extent, this paper analyzes the major IoT industrial platforms fulfilling three main concerns. First, it describes the selected platforms, giving the reader enough details to understand the differences among them. Second, it compares them against the extracted security criteria, picturing the overall security level of the different proposals. Finally, it gives a thorough example of an IoT service implementation on two previously analyzed platforms, providing a solid guideline to researchers and developers who want to contribute in this direction.

Regarding the future of IoT, we think the implementation and popularization of new IoT services in new spaces where they have not been seen before will occur [58]. In contexts such as smart city, environment, physical security, retailing or agriculture, new applications will emerge such as smart roads that provide useful information about accidents or road conditions, earthquake detection systems that report anomalous earth movements, solutions for detection of harmful gases in inhabited spaces, recommendation of products for a user according to his reported allergies and preferences and modules that control micro climates and its impact on production. We also believe that the application of IoT services in new contexts can require some customization or adaptation that will also provoke the development of new IoT architectures, therefore new IoT platforms such as those mentioned in this paper that fit specific security requirements, integrating heterogeneous IoT services to compose a System of Systems (SoS) [59] and guaranteeing always affordable business models. As new IoT services are integrated to a greater extent to our society, our technological dependency will also increase and the impact of a security incident can be more disastrous, motivating a continuous research around new cyber threats and countermeasures.

Future works will explore the possibility of adding more platforms to the selected ones. Moreover, we will investigate on proposing a parameter to measure the security level of the IoT platforms.

Author Contributions: Funding acquisition, F.G.M.; Investigation, D.D.L., M.B.U., C.S.C., D.T.M., E.G.G., P.N. and F.G.M.; Writing—original draft, D.D.L., M.B.U., C.S.C., D.T.M., E.G.G. and P.N.; and Writing—review and editing, D.D.L. and F.G.M.

Funding: This work was funded by a Ramón y Cajal research contract (RYC-2015-18210) granted by the Ministerio de Economía y Competitividad (Spain) and co-funded by the European Social Fund, a Leonardo Grant 2017 for Researchers and Cultural Creators awarded by Fundación BBVA, and Escuela Colombiana de Ingeniería Julio Garavito (Colombia) through the project "Cyber Security Architecture for Incident Management", approved by the Internal Research Opening 2017.

Conflicts of Interest: The authors declare no conflict of interest. The funding sponsors had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, and in the decision to publish the results.

References

1. Gartner. *Gartner's 2016 Hype Cycle for Emerging Technologies Identifies Three Key Trends That Organizations Must Track to Gain Competitive Advantage*; Gartner: Stamford, CT, USA, 2016.
2. Li, S.; Da Xu, L.; Zhao, S. *The Internet of Things: A Survey*; Springer: New York, NY, USA, 2015; pp. 243–259.
3. Yelamarthi, K.; Aman, M.S.; Abdelgawad, A. An application-driven modular IoT architecture. *Wirel. Commun. Mob. Comput.* **2017**, *2017*, 1350929. [CrossRef] [PubMed]
4. Gomes, B.; Muniz, L.; da Silva e Silva, F.J.; Ríos, L.E.T.; Endler, M. A comprehensive cloud-based IoT software infrastructure for Ambient Assisted Living. In Proceedings of the 2015 International Conference on Cloud Technologies and Applications (CloudTech), Marrakech, Morocco, 2–4 June 2015; pp. 1–8.

5. Yaqoob, I.; Ahmed, E.; Hashem, I.A.T.; Ahmed, A.I.A.; Gani, A.; Imran, M.; Guizani, M. Internet of Things Architecture: Recent Advances, Taxonomy, Requirements, and Open Challenges. *IEEE Wirel. Commun.* **2017**, *24*, 10–16. [CrossRef]
6. Połap, D.; Kęsik, K.; Książek, K.; Woźniak, M. Obstacle Detection as a Safety Alert in Augmented Reality Models by the Use of Deep Learning Techniques. *Sensors* **2017**, *17*, 2803. [CrossRef] [PubMed]
7. Woźniak, M.; Połap, D. Object detection and recognition via clustered features. *Neurocomputing* **2018**, *320*, 76–84. [CrossRef]
8. Zarpelo, B.B.; Miani, R.S.; Kawakani, C.T.; de Alvarenga, S.C. A Survey of Intrusion Detection in Internet of Things. *J. Netw. Comput. Appl.* **2017**, *84*, 25–37. [CrossRef]
9. Abdelgawad, A.; Yelamarthi, K. Internet of things (IoT) platform for structure health monitoring. *Wirel. Commun. Mob. Comput.* **2017**, *2017*, 6560797. [CrossRef]
10. Połap, D.; Winnicka, A.; Serwata, K.; Kęsik, K.; Woźniak, M. An Intelligent System for Monitoring Skin Diseases. *Sensors* **2018**, *18*, 2552. [CrossRef] [PubMed]
11. Charmonman, S.; Mongkhonvanit, P. Special consideration for Big Data in IoE or Internet of Everything. In Proceedings of the 2015 13th International Conference on ICT and Knowledge Engineering (ICT Knowledge Engineering 2015), Bangkok, Thailand, 18–20 November 2015; pp. 147–150.
12. Wollschlaeger, M.; Sauter, T.; Jasperneite, J. The Future of Industrial Communication: Automation Networks in the Era of the Internet of Things and Industry 4.0. *IEEE Ind. Electron. Mag.* **2017**, *11*, 17–27. [CrossRef]
13. Ju, J.; Kim, M.S.; Ahn, J.H. Prototyping Business Models for IoT Service. *Procedia Comput. Sci.* **2016**, *91*, 882–890. [CrossRef]
14. Gupta, H.; Vahid Dastjerdi, A.; Ghosh, S.K.; Buyya, R. iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments. *Softw. Pract. Exp.* **2017**, *47*, 1275–1296. [CrossRef]
15. Sarkar, S.; Chatterjee, S.; Misra, S. Assessment of the Suitability of Fog Computing in the Context of Internet of Things. *IEEE Trans. Cloud Comput.* **2018**, *6*, 46–59. [CrossRef]
16. Nespoli, P.; Papamartzivanos, D.; Gómez Mármol, F.; Kambourakis, G. Optimal Countermeasures Selection against Cyber Attacks: A Comprehensive Survey on Reaction Frameworks. *IEEE Commun. Surv. Tutor.* **2018**, *20*, 1361–1396. [CrossRef]
17. Al-Fuqaha, A.; Guizani, M.; Mohammadi, M.; Aledhari, M.; Ayyash, M. Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. *IEEE Commun. Surv. Tutor.* **2015**, *17*, 2347–2376. [CrossRef]
18. Xu, L.D.; He, W.; Li, S. Internet of Things in Industries: A Survey. *IEEE Trans. Ind. Inform.* **2014**, *10*, 2233–2243. [CrossRef]
19. Perera, C.; Liu, C.H.; Jayawardena, S. The Emerging Internet of Things Marketplace From an Industrial Perspective: A Survey. *IEEE Trans. Emerg. Top. Comput.* **2015**, *3*, 585–598. [CrossRef]
20. Ganguly, P. Selecting the right IoT cloud platform. In Proceedings of the 2016 International Conference on Internet of Things and Applications (IOTA), Pune, India, 22–24 January 2016; pp. 316–320.
21. Guth, J.; Breitenbücher, U.; Falkenthal, M.; Leymann, F.; Reinfurt, L. Comparison of IoT platform architectures: A field study based on a reference architecture. In Proceedings of the 2016 Cloudification of the Internet of Things (CIoT), Paris, France, 23–25 November 2016; pp. 1–6.
22. Derhamy, H.; Eliasson, J.; Delsing, J.; Priller, P. A survey of commercial frameworks for the Internet of Things. In Proceedings of the 2015 IEEE 20th Conference on Emerging Technologies Factory Automation (ETFA), Luxembourg, 8–11 September 2015; pp. 1–8.
23. Yaqoob, I.; Ahmed, E.; ur Rehman, M.H.; Ahmed, A.I.A.; Al-garadi, M.A.; Imran, M.; Guizani, M. The rise of ransomware and emerging security challenges in the Internet of Things. *Comput. Netw.* **2017**, *129*, 444–458. [CrossRef]
24. Ammar, M.; Russello, G.; Crispo, B. Internet of Things: A survey on the security of IoT frameworks. *J. Inf. Secur. Appl.* **2018**, *38*, 8–27. [CrossRef]
25. Díaz López, D.; Blanco Uribe, M.; Santiago Cely, C.; Vega Torres, A.; Moreno Guataquira, N.; Morón Castro, S.; Nespoli, P.; Gómez Mármol, F. Shielding IoT against cyber-attacks: An event-based approach using SIEM. *Wirel. Commun. Mob. Comput.* **2018**, *2018*, 3029638. [CrossRef]
26. Beltran, V.; Skarmeta, A.; Ruiz, P. An ARM-Compliant Architecture for User Privacy in Smart Cities: SMARTIE—Quality by Design in the IoT. *Wirel. Commun. Mob. Comput.* **2017**, *2017*, 3859836. [CrossRef]

27. Ferrag, M.A.; Maglaras, L.A.; Janicke, H.; Jiang, J.; Shu, L. Authentication Protocols for Internet of Things: A Comprehensive Survey. *Secur. Commun. Netw.* **2017**, *2017*, 6562953. [CrossRef]
28. Nespoli, P.; Zago, M.; Huertas Celdrán, A.; Gil Pérez, M.; Gómez Mármol, F.; García Clemente, F.J. A Dynamic Continuous Authentication Framework in IoT-Enabled Environments. In Proceedings of the Fifth International Conference on Internet of Things: Systems, Management and Security (IoTSMS 2018), Valencia, Spain, 15–18 October 2018.
29. Boneh, D.; Sahai, A.; Waters, B. Functional encryption: Definitions and challenges. In *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*; LNCS: Berlin, Germany, 2011; Volume 6597, pp. 253–273.
30. Saxena, N.; Choi, B.J.; Lu, R. Authentication and Authorization Scheme for Various User Roles and Devices in Smart Grid. *IEEE Trans. Inf. Forensics Secur.* **2016**, *11*, 907–921. [CrossRef]
31. Goldstein, J.; Pagan, F.; Short, J. Systems and Methods for Providing Dynamic Network Authorization Authentication and Accounting. Patent EP20000973771, 20 October 2000.
32. Sforzin, A.; Gómez Mármol, F.; Conti, M.; Bohli, J.M. RPiDS: Raspberry Pi IDS—A Fruitful Intrusion Detection System for IoT. In Proceedings of the 13th IEEE International Conference on Advanced and Trusted Computing (ATC 2016), Toulouse, France, 18–21 July 2016; pp. 440–448.
33. Useche Peláez, D.; Díaz López, D.; Nespoli, P.; Gómez Mármol, F. TRIS: A Three-Rings IoT Sentinel to protect against cyber-threats. In Proceedings of the Fifth International Conference on Internet of Things: Systems, Management and Security (IoTSMS 2018), Valencia, Spain, 15–18 October 2018.
34. Hunke, N.; Rüßmann, M.; Schmieg, F.; Bhatia, A.; Kalra, N. Winning in IoT: It's All about the Business Processes. Available online: https://www.bcg.com/en-co/publications/2017/hardware-software-energy-environment-winning-in-iot-all-about-winning-processes.aspx (accessed on 25 October 2018).
35. Lucero, S. *IoT Platforms: Enabling the Internet of Things*; IHS Technology: Phoenix, AZ, USA, 2016.
36. Wootton, C. *Samsung ARTIK Reference: The Definitive Developers Guide*; Apress: New York, NY, USA, 2016; p. 409.
37. Wootton, C. *Beginning Samsung ARTIK—A Guide to Developers*; Apress: New York, NY, USA, 2016; p. 396.
38. Kurniawan, A. *Learning AWS IoT: Effectively Manage Connected Devices on the AWS Cloud Using Services Such as AWS Greengrass, AWS Button, Predictive Analytics and Machine Learning*; Packt Publishing Ltd.: Birmingham, UK, 2018; p. 278.
39. Tarneberb, W.; Chandrasekaran, V.; Humpherey, M. Experiences Creating a Framework for Smart Traffic Control using AWS IoT. In Proceedings of the 2016 ACM 9th International Conference on Utility and Cloud Computing, Shanghai, China, 6–9 December 2016; pp. 63–69.
40. Azraq, A.; Chughtai, S.; Mashhour, A.; V Nguyen, D.; Dos Santos, R.M. *Enhancing the IBM Power Systems Platform with IBM Watson Services*; IBM Redbooks: New York, NY, USA, 2018; p. 218.
41. Ravulavaru, A. *Enterprise Internet of Things Handbook: Build End-to-End IoT Solutions Using Popular IoT Platforms*; Packt Publishing: Birmingham, UK, 2018.
42. PratimRay, P. A survey of IoT cloud platforms. *Future Comput. Inform. J.* **2016**, *1*, 35–46.
43. Vossen, G.; Schonthaler, F.; Dillon, S. *The Web at Graduation and Beyond: Business Impacts and Developments*; Springer: New York, NY, USA, 2016.
44. EVRYTHNG IoT Smart Products Platform. Available online: https://evrythng.com/ (accessed on 12 July 2018).
45. Guinard, D.; Trifa, V. *Building the Web of Things*; Manning Publications Co.: Greenwich, CT, USA, 2016.
46. Dweet—Data Sharing for IoT. Available online: https://dweet.io/ (accessed on 12 September 2018).
47. Freeboard—Dashboards for the Internet of Things. Available online: https://freeboard.io/ (accessed on 1 September 2018).
48. Blackstock, M.; Lea, R. Toward a Distributed Data Flow Platform for the Web of Things (Distributed Node-RED). In Proceedings of the 5th International Workshop on Web of Things (WoT '14), Cambridge, MA, USA, 8 October 2014; pp. 34–39.
49. Yasumoto, K.; Yamaguchi, H.; Shigeno, H. Survey of Real-Time Processing Technologies of IoT Data Streems. *J. Inf. Process.* **2016**, *24*, 195–202.
50. Kamal, R. *Internet of Thinks: Architecture and Design Principles*; McGraw Hill Education: New York, NY, USA, 2017.

51. Kocovic, P.; Behringer, R.; Ramachandran, M.; Mihajlovic, R. *Emerging Trends and Applications of the Internet of Things*; IGI Global: Hershey, PA, USA, 2017.
52. Minerauda, J.; Mazhelisb, O.; Suc, X.; Tarkomaa, S. A gap analysis of Internet-of-Things platforms. *Comput. Commun.* **2016**, *89–90*, 5–16. [CrossRef]
53. Mavromoustakis, C.X.; Mastorakis, G.; Dobre, C. *Advances in Mobile Cloud Computing and Big Data in the 5G Era*; Springer: New York, NY, USA, 2016.
54. Martino, B.D.; Li, K.C.; Yang, L.T.; Esposito, A. *Internet of Everything: Algorithms, Methodologies, Technologies and Perspectives*; Springer: New York, NY, USA, 2017.
55. Geng, H. *Internet of Things and Data Analytics Handbook*; John Wiley & Sons: New York, NY, USA, 2017.
56. Spitzer, C.; Ferrell, U.; Ferrell, T. *Digital Avionics Handbook*, 3rd ed.; CRC Press: Boca Raton, FL, USA, 2017.
57. Soro, A.; Ambe, A.H.; Brereton, M. Minding the Gap: Reconciling Human and Technical Perspectives on the IoT for Healthy Ageing. *Wirel. Commun. Mob. Comput.* **2017**, *2017*, 7439361. [CrossRef]
58. Cao, T.D.; Hoang, H.H.; Huynh, H.X.; Nguyen, B.M.; Pham, T.V.; Tran-Minh, Q.; Tran, V.T.; Truong, H.L. Iot services for solving critical problems in vietnam: A research landscape and directions. *IEEE Internet Comput.* **2016**, *20*, 76–81. [CrossRef]
59. Thacker, S.; Pant, R.; Hall, J.W. System-of-systems formulation and disruption analysis for multi-scale critical national infrastructures. *Reliab. Eng. Syst. Saf.* **2017**, *167*, 30–41. [CrossRef]