Canella: Privacy-Aware End-to-End Integrated IoT Development Ecosystem

Atheer Aljeraisy
School of Computer Science and Informatics
Cardiff University
Cardiff, UK
Majmaah University, Saudi Arabia
aljeraisya@cardiff.ac.uk

Omer Rana
School of Computer Science
Cardiff University
Cardiff, UK
ranaof@cardiff.ac.uk

Charith Perera
School of Computer Science
Cardiff University
Cardiff, UK
PereraC@cardiff.ac.uk

***Abstract*—Applications for the Internet of Things (IoT) can derive sensitive information about people, so developers must protect users' privacy in compliance with privacy and data protection laws. However, developers face difficulties in addressing privacy issues as many applications exploit personal data in a problematic manner. We present Canella, an integrated IoT development ecosystem that is augmented with novel privacy-preserving components for developing privacy-aware IoT applications. Canella helps software developers meet privacy requirements during the development phase and during rapid prototyping, and it provides real-time feedback on potential privacy concerns. Canella aims at assisting developers to (i) better understand their code's behavior, (ii) better overcome privacy issues and comply with privacy and data protection laws, (iii) reduce time to incorporate privacy into an IoT application, and (iv) reduce the cognitive load of integrating privacy into an IoT application. Thus, Canella can result in a significant improvement in building privacy-aware IoT applications. (Demo Video)**

***Index Terms*—Internet of Things, Privacy by Design, Software Development, Data Protection, Privacy Law, Usable Privacy**

I. Introduction and Motivation

Applications for the Internet of Things (IoT) can derive sensitive information about individuals, so developers must ensure that data is maintained in accordance with privacy and data protection legislation. Nevertheless, developers still face many difficulties in handling privacy issues, leading to numerous applications exploiting sensitive data in a problematic way for a number of reasons. First, it has been found that developers do not treat privacy as a primary concern [1]. Second, developers have a partial understanding of what should be considered to protect users' data [2]. Finally, developers lack knowledge of their apps' data practices, which makes it difficult to design and implement effective privacy requirements [3].

Recent studies have revealed that developers face challenges when complying with privacy and data protection laws when the regulations' terms are ambiguous and it is hard to understand and apply these legal requirements [4]. Due to this, it becomes important to transform these privacy regulations into software requirements, a process known as Privacy by Design (PbD) [5]. Several researchers have proposed PbD principles, guidelines, strategies, and patterns to assist developers in effectively integrating privacy-preserving techniques into the development process [4]. The General Data Protection Regulation (GDPR) [4] assures the importance of PbD and its applicability to all systems that process personal data, a characteristic of IoT applications. However, most guidelines are legal rather than technical. This makes them unsuitable for application-building tools, disconnecting them from developers' practical environment. Hence, there is a need to enrich existing developers' tools with privacy-preserving components. This will increase developers' awareness of how personal information is utilized and misused [4].

In this paper, we present Canella, an integrated IoT development ecosystem. It is augmented with novel privacy-preserving components that work together to help developers build privacy-friendly end-to-end IoT applications. Canella provides real-time feedback on potential privacy concerns and recommends privacy-preserving components to be integrated into IoT data flows. This can encourage developers to consider the potential effects of their applications and the possibility of using less risky options when dealing with personal data. As a result, this can reduce developers' cognitive load and the time it takes to incorporate privacy into an IoT application.

II. Focus Group for Requirements Gathering

As the PbD schemes are the mechanism to comply with privacy and data protection laws, Aljeraisy et al. have mapped the elements of the PbD schemes with the principles and individuals' rights of their Combined Privacy Law Framework (CPLF)[1]. Thus, we conducted a focus group study with novice developers to explore the applicability and feasibility of designing and implementing these PbD schemes into different IoT development environments, such as Arduino, Blockly, and Node-RED, in a reusable manner. The participants were divided into 25 groups, with three members in each group.

The results of the focus group study led us to design and implement the privacy-preserving components of Canella based on the PbD guidelines suggested by Perera et al. [6]. This could refer to the clarity of the specifications of these guidelines in comparison to other elements of the PbD schemes as well as their context as they are specific to the IoT domain. After analyzing all participants' responses, we

[1]The Combined Privacy Laws Framework refers to the selected privacy and data protection laws analyzed in Aljeraisy et al.'s study [4].

excluded the guidelines that were not properly supported by useful feedback. The resultant guidelines are shown in Figure 1. Each guideline complies with one or more of the CPLF principles. For example, a developer will follow the CPLF's Data Minimization principle when integrating Minimize Data Storage into an IoT application. In this paper, we decided to focus on the guidelines that are in line with the Data Minimization principle of the CPLF as a starting point for designing and implementing privacy-preserving components. These are as follows: (1) Minimize Raw Data Intake; (2) Minimize Data Storage; (3) Reduce Location Granularity; and (4) Category-Based Aggregation.

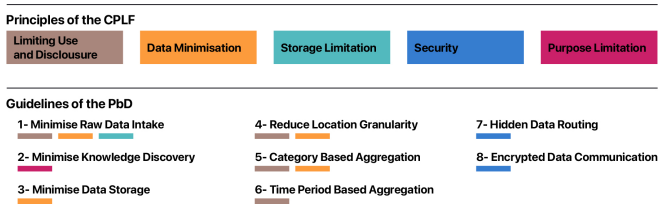

Fig. 1. The Relationship between Applying these PbD Guidelines and Compliance with the Principles of the CPLF.

III. Design and Implementation of Canella

A. Architecture of Canella

The architecture of Canella is based on the IoT architecture in terms of how data moves through an IoT application. Typically, data moves from sensing devices to gateway devices and then to the cloud infrastructure. This pattern is usually comprised of three components, which are edge nodes, fog nodes, and cloud nodes, and these all have different computational capabilities. As shown in Figure 2, Canella utilizes two widely used community IoT development tools:

Blockly—is an open-source library developed by Google for adding block-based visual programming to an application. The Blockly editor provides a user interface and a framework to generate code using interlocking and graphical blocks [7].

Node-RED—is an open-source flow-based visual programming development tool originally developed by IBM for integrating hardware devices, APIs, and online services. It provides a browser-based flow editor to (1) drag, drop, and connect nodes in the pallets or (2) import JavaScript code [8].

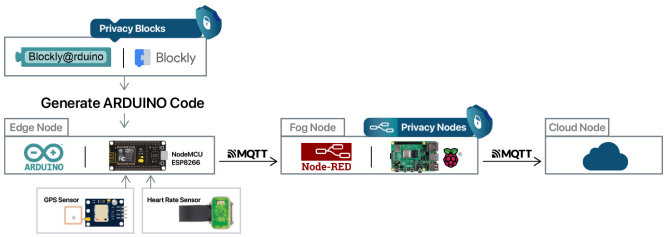

Fig. 2. Canella's Architecture.

Figure 2 depicts Canella's architecture, which particularly utilizes Blockly@rduino [9]. It is a visual programming editor based on Google's Blockly that has been forked to generate Arduino's C/C++ code. The Blockly@rduino represents the edge of an IoT application in Canella, using a microcontroller (NodeMCU ESP8266). The Node-RED is the fog of an IoT application in Canella that performs data processing and storage. It is based on a single-board microcomputer with complex computational capabilities (the Raspberry Pi). The Node-RED can directly access data from sensors connected to an Arduino board in several ways (e.g., via a serial port number or WiFi using a protocol called MQTT). Then, the Node-RED sends the analytical information to the cloud through WiFi using the MQTT protocol for storage purposes.

Canella's goal is to help developers handle privacy issues on the edge and fog nodes of an IoT app to reduce privacy risks before data is sent to the cloud node. As a result, we created ***privacy blocks*** in Blocky@rduino and ***privacy nodes*** in Node-RED (Figure 2). Together, these privacy-preserving components help IoT developers meet privacy requirements throughout the data lifecycle when building IoT applications.

B. Design and Implementation of Privacy Components

In this section, we demonstrate the design and implementation of the PbD guidelines. In particular, each ***privacy block*** and ***privacy node*** corresponds to a specific PbD guideline that is in line with CPLF principles. We define the shape of a block, its field, and its connection points. Likewise, we specify the node's properties, edit dialog, and help text. To assist developers in protecting personal data without compromising the IoT application objective, we suggest different choices for users' data values. ***Privacy blocks*** and ***privacy nodes*** are designed to be generic enough to be used in any IoT application. Canella provides detailed information for developers about the objective of each ***privacy block*** and ***privacy node***. In the same way, it supports information about compliance with privacy and data protection laws on each ***privacy block*** and ***privacy node*** to raise developers' awareness of privacy and data protection regulations.

Reduce Location Granularity: Granularity refers to the level of detail represented by the data. High granularity refers to detail at the atomic level, while low granularity zooms out into a summary. The dissemination of location can be considered coarse-grained, while the full address can be considered fine-grained, which poses more privacy risks. The Reduce Location Granularity performs reverse geocoding to convert longitude and latitude into a human-readable address. Accordingly, we designed its block and node to suggest to developers three GPS coordinates for reducing location granularity to a postcode, city name, or country name, requiring an API key to access the Google Maps API (Figure 3).

Minimize Raw Data Intake: IoT apps should convert raw data into secondary context data to avoid privacy violations. We designed the Minimize Raw Data Intake block to reduce the amount of raw data it receives by taking the average of the sensor data values over a specified period of time. Developers can choose different types of data to minimize as well as units of time (seconds, minutes, and hours) and are required to input a number value for a selected time to calculate the average of the data (Figure 3). In Node-RED, however, it only reduces the amount of raw data by calculating the average