Developing GDPR Compliant Apps For The Edge *

Tom Lodge[1][0000−0003−0857−7341], Andy Crabtree[1][0000−0001−5553−6767], and Anthony Brown[2][0000−0001−5465−8009]

[1] School of Computer Science, University of Nottingham, UK
{thomas.lodge,andrew.crabtree}@nottingham.ac.uk
[2] Horizon Digital Economy Research, University of Nottingham, UK
anthony.brown@nottingham.ac.uk

Abstract. We present an *overview* of the Databox application development environment or SDK as a means of enabling trusted IoT app development at the network edge. The Databox platform is a dedicated domestic platform that stores IoT, mobile and cloud data and executes local data processing by third party apps to provide end-user control over data flow. Key challenges for building apps in edge environments concern *(i)* the complexity of IoT devices and user requirements, and *(ii)* supporting privacy preserving features that meet new data protection regulations. We examine how the Databox SDK can ease the burden of regulatory compliance and be used to sensitize developers to privacy related issues in the very course of building apps.

Keywords: Internet of Things · edge computing · Databox · data protection · GDPR · trusted application development · SDK.

1 Introduction

The predominant paradigm for computing is centred in the cloud. However, as the Internet of Things (IoT) emerges, the requirement to push increasing volumes of data to the network for centralized storage and processing will impact system resilience, network traffic, latency and privacy. An alternative approach is to "extend the cloud to where things are" [1] and shift data storage and processing to the edge of the network. In this model, nodes at the edge perform the bulk of storage and processing, keeping data off the core network, reducing latency and improving the potential for data privacy. The model has gained significant traction in recent years, and the IDC [2] predicts investment in edge infrastructure will reach up to 18% of total spend by 2020.

The domestic space is seeing a growth in dedicated hardware that brings more data storage and processing to the edge [3–7]. Many of these products unify access to connected home devices and provide facilities (voice, web UIs,

* This work was supported by the Engineering and Physical Sciences Research Council (Grant Numbers EP/M001636/1, EP/N028260/1, EP/M02315X/1).

apps) for automation and control. With new General Data Protection Regulation (GDPR) in Europe [8], and growing concern amongst ordinary people about the (ab)use of personal data, we anticipate this space will grow to include new domestic platforms that take a more principled approach to exploiting personal data generated by IoT devices, mobile and cloud services. The Databox platform [9] provides one of several [10–12] instantiations of domestic 'privacy-preserving' edge-based solutions, running data processors (apps) within a sandboxed environment where access to and use of data is constrained by user-negotiated contracts.

The distinguishing feature of such platforms is that processing moves to the data, rather than data to the processing, and data distribution is limited to the results of local queries enabling the 'data minimisation' that is required under GDPR. Developing apps that run on these platforms is challenging. There are challenges that are already familiar to IoT developers: *(i)* processing data from an increasingly heterogeneous range of data sources, *(ii)* across a wide variability of domestic environments and *(iii)* competing systems with inconsistent patterns of behaviour [13], plus *(iv)* the need to support multiple users with diverse requirements. There are also new challenges that come from the need to meet new data protection regulation and (thereby) gain user trust. This requires that developers demonstrably respond to the requirements of data protection regulation in the apps they produce [14]. Moreover 'developers' is a broad category including makers, hobbyists, and enthusiasts. Development environments must therefore enable data protection across a broad cohort while providing developers and end-users alike with the tools they need to build the (often niche) functionality that they require.

Our end-user development environment (SDK) has been designed to build apps for the Databox platform and to: *(i)* simplify IoT app development for domestic environments, in particular data processing across multiple devices and sensors; *(ii)* open up development to a broad cohort of developers and *(iii)* enable compliance with key features of GDPR. Though our SDK addresses all of these challenges, this paper focuses exclusively upon *(iii)*, i.e. how developers can be supported when creating domestic IoT privacy preserving apps that are compliant with the letter and spirit of GDPR.

This paper has two main contributions: *(i)* an assessment of the implications of GDPR upon the creation of edge-based personal data processing systems *(ii)* design and implementation of a development environment for building GDPR compliant domestic apps. This latter contribution has relevance beyond a description of design and implementation choices; it points to a new set of general features we expect will be of value to any development environment geared towards writing code that operates upon personal data.

2 Related Work

We briefly consider 3 interconnected areas of work: *(i)* domestic smart hubs, *(ii)* privacy preserving environments and *(iii)* developer support.

2.1 Domestic Smart Hubs

The multitude of different standards, network and data protocols employed within the domestic IoT space has resulted in the emergence of IoT ecosystems aimed at providing *(i)* interoperability across devices *(ii)* control interfaces for device management, and *(iii)* support for home automation. Within the open source community, many IoT systems have also been designed to run on local hardware, whether ARM, x86 or embedded system such as Arduino and Raspberry Pi [15–18]. These systems are aimed at technically competent users and are underpinned by programming frameworks to support further extension.

There is also a highly competitive startup scene, with a range of products on the market aimed at the general consumer [19–21], typically offering easy integration with IoT devices and polished control interfaces. The most significant inroads have been made by the large Internet companies. Amazon's 'Echo' [3] is installed in tens of millions of households, for example, and Google's 'Home' [5] is gaining market share as is Apple's HomePod [4]. These systems perform some local storage and processing as a means of reducing latency and reliance on an upstream network, but still use companion cloud-based systems when needed. However, the mechanisms and processes utilised by these cloud systems remain opaque to the end user. Not only is there a lack of transparency around the flow and use of data, there are notably few features enabling users to restrict data flow or exploit it for individual purposes.

2.2 Privacy Preserving Environments

Personal Data Management Services, whether cloud-based [22] or at the edge [23] store consumer data and provide explicit contracts to underpin data exchange.

The Databox platform is a privacy preserving domestic smart hub that permits controlled access to a data subject's personal data, set out in explicit user-agreed contracts called Service Level Agreements (SLAs). The system provides abstractions for data sources (IoT devices or cloud-based services such as Twitter), drivers (privileged code that communicates with datasources), datastores (local repositories of user data) and apps (code that processes data). Apps are untrusted code, and can only ever communicate with datastores (to read data or actuate a device) with explicit consent from a user. All components (including apps) run in isolated Docker[3] containers. Restrictions are enforced through an arbiter. Fig.1 *(1,2 and 3)* shows the token exchange. At app install time the SLA is parsed, and permissions granted (the arbiter is informed app X can do action Y). Tokens are not minted until the app requests one (usually just before it performs an action). Tokens have expiry dates and can be cached and reused until expiry, after which a new one must be requested. The wider Databox ecology consists of an app store; a repository of databox apps that can be downloaded to an individual Databox, and an SDK; a web-based development environment for constructing apps. Users interact with the Databox through a web frontend,

[3] https://www.docker.com

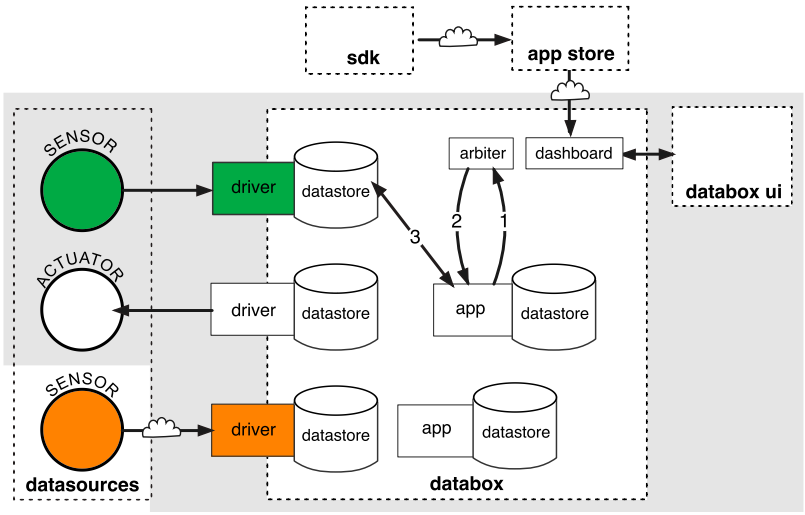

Fig. 1. Databox platform architecture

which provides a set of interfaces for installing new apps (part of which will require users to review the app contract) and to view/monitor/remove running apps. The platform is responsible for auditing all accesses to datastores and enforcing SLAs.

2.3 Developer support

The matter of developer support for IoT hubs is not straightforward. Commercial and open source ecosystems provide development environments that support the creation of new product integrations or bespoke functionality oriented around a product's features [24–27] and are typically targeted at competent and/or professional programmers. However, Newman [28] has noted that the burgeoning array of connected domestic devices makes it intractable for developers to build applications to keep pace with the needs of users. He thus argues for the need to support end-user programming to allow a diverse cohort of people to *"compose the functionality that they need"*. Perhaps as a result of these observations, we have seen a proliferation of graphical end-user programing environments [29–32] aimed at masking device/service/protocol heterogeneity and helping connect IoT and webservices in new and interesting ways. The most popular, IFTTT, enjoys a considerable user base [33]. However, given the focus upon technical simplicity, privacy preserving features are given scant regard. Indeed [34] found that 50% of the nearly 20,000 IFTTT 'recipes' they examined contained secrecy or integrity violations that could lead to harm.

3 GDPR compliance and its influence on developers

In GDPR a data controller *"determines the purposes and means of processing personal data"*. When developing applications that run upon IoT hubs, if app developers receive personal data, they are controllers. Similarly, if developers create the app on behalf of a third party they must demonstrate 'privacy by design' principles. Development environments, therefore, must take this into account. Article *5(2)* states: *"the controller shall be responsible for, and be able to demonstrate, compliance with the principles [of GDPR]"*.

In working through the regulation we posit that IoT app developers are implicated in two broad areas: *(i)* transparency and *(ii)* articulating and appropriately reducing risk. GDPR explicitly mentions a requirement for risk assessment in Article *35* (data protection impact assessments), though the mention of risk and mechanisms for its reduction are sprinkled throughout various clauses. Article *25 (1)* explicitly requires risk assessment and reduction is performed *"at the time of the determination of the means for processing"*, i.e. at app development time.

GDPR's risk concerns are oriented around data disclosure and automated profiling. Other risks such as physical risk (e.g. switching on an empty kettle, closing an automatic garage door), fall outside its scope, though clearly must be given due consideration by developers. Automated profiling relates to harms from unfair, inaccurate algorithmic decisions (whether deliberate or unintentional) that have socially consequential outcomes (e.g. denial of credit / employment / healthcare). This is a burgeoning area of research [35–37] and we have begun early exploration with two new features in our SDK (see our special purpose profiling node and runtime inspection interface in Section 4).

Transparency relates to adequate provision of information relating to the collection, processing and use of personal data in order that users have information to *(i)* provide informed consent and *(ii)* control (restrict, extend, halt) its use. Transparency is in itself advocated in GDPR as a tool to reduce risk, and many of the basic "rights" enshrined by the regulation are predicated upon it, i.e. the right to object, the right to be informed and the right to restrict processing.

When considering the impact of GDPR upon developers, we assume the platform (i.e. IoT hub, such as Databox) will take most responsibility for data security, notification of breaches, ongoing data storage and access (Articles *5, 16, 17, 20, 25, 30, 32-34*). That is not to disregard their importance or to suggest that the developer can be disconnected from these concerns, only that they sit outside the scope of this work.

Given this scope, Tables 1a and 1b distil the 99 key parts of the Articles (*5, 7, 12, 13, 21, 22, 25, 35)* that implicate developers with regard to data disclosure risk and/or transparency requirements. The 3rd column *(R/T)* marks each clause as either relating to risk (R) or transparency (T). Note that for the sake of brevity we do not include Article *13's* list of information to be provided; interested readers are directed to Article *13(1)* in the full text [8]. A few clauses remain open to interpretation and have garnered considerable debate in the legal and academic communities. Nevertheless, it is our view that the spirit of GDPR is clear and developer tools have a necessary role in helping meet requirements.

Table 1a. GDPR clauses relevant to developers

Art	Relevant Clauses	R/T
5	(a) processed lawfully, fairly and in a transparent manner in relation to the data subject ('lawfulness, fairness and transparency');	*T*
	(b) collected for specified, explicit and legitimate purposes and not further processed in a manner that is incompatible with those purposes;	*T*
	(c) adequate,relevant and limited to what is necessary in relation to the purposes for which they are processed ('data minimisation')	*R*
7	4. Where processing is based on consent, the controller shall be able to demonstrate that the data subject has consented to processing of his or her personal data.	*T*
	8. The data subject shall have the right to withdraw his or her consent at any time. [. . .] It shall be as easy to withdraw consent as to give it.	*T*
12	1. provide any information [. . .] relating to processing to the data subject in a concise, transparent, intelligible and easily accessible form, using clear and plain language	*T*
	7. The information to be provided [..] may be provided in combination with standardised icons in order to give in an easily visible, intelligible and clearly legible manner a meaningful overview of the intended processing.	*T*
13	2 (f) the existence of automated decision-making, including profiling, referred to in Article 22(1) and (4) and, at least in those cases, meaningful information about the logic involved, as well as the significance and the envisaged consequences of such processing for the data subject.	*T*
21	1. The data subject shall have the right to object, on grounds relating to his or her particular situation, at any time to processing of personal data concerning him or her which is based on points (e) or (f) of Article 6(1), including profiling based on those provisions.	*T*
22	1. The data subject shall have the right not to be subject to a decision based solely on automated processing, including profiling, which produces legal effects concerning him or her or similarly significantly affects him or her	*R*
	2. Paragraph 1 shall not apply if the decision is: c) based on the data subject's explicit consent	*T*

Table 1b. GDPR clauses relevant to developers, ctd.

Art	Relevant Clauses	R/T
25	1. [..] the controller shall, both at the time of the determination of the means for processing and at the time of the processing itself, implement appropriate technical and organisational measures, such as pseudonymisation, [..] such as data minimisation.	*R*
	2. The controller shall implement appropriate technical and organisational measures for ensuring that, by default, only personal data which are necessary for each specific purpose of the processing are processed.	*R*
35	1. Where a type of processing in particular using new technologies, and taking into account the nature, scope, context and purposes of the processing, is likely to result in a high risk to the rights and freedoms of natural persons, the controller shall, prior to the processing, carry out an assessment of the impact of the envisaged processing operations on the protection of personal data.	*R*

4 The Databox SDK

The SDK is a fully featured web-based environment for building Databox apps. It provides facilities for testing, tools for data visualisation, context-sensitive help, skeleton code generation, basic static type checking and code management (Fig.2).

The SDK models apps as information flows (inspired by the flow-based programming paradigm [38]) and abstracts the Databox platform architecture into four 'node' types: *datastores, processors, profilers and outputs. Datastores* represent all devices (or services) that generate data. Datastores are device independent, i.e. a smart plug datastore will present a consistent data schema in the SDK, independent of the specific device or manufacturer it maps to at runtime. *Processor* nodes operate on data; it is here custom behaviours and logic are encoded. Processor nodes typically consume one or more inputs and send results to one or more outputs. *Profiler* nodes are a special category of processing node that *infer new information* about a data subject. In treating profilers differently from processing nodes, we aim (in subsequent iterations of the SDK) to sensitise developers to GDPR's more restrictive covenants around "automated profiling" by providing facilities to assess the fairness of profiling on target users [35]. *Output* nodes perform an action, such as actuation, visualization, or data export.

When developers publish an app from the SDK, they are prompted for information needed to construct the SLA (user-negotiated contract). Once deployed, the app's datasource nodes interact with the Databox platform API to request permissions to access data according to the terms of the SLA. This functionality is transparently provided by the SDK, insulating developers from the detail.

Fig. 2. Inputs, processors, profilers and outputs

4.1 Features Enabling GDPR Compliance

In Section 3 we presented the set of GDPR clauses that will implicate developers building apps for domestic IoT hubs. We scoped the problem into *(i)* assessing and reducing disclosure risk and *(ii)* transparency on what/how/why personal data is being processed.

Our SDK sensitizes developers to data disclosure risk by *(i)* providing ongoing risk breakdowns as developers build apps *(ii)* tracking personal data as it moves through an app. Our SDK addresses GDPRs requirements for transparency by *(i)* creating GDPR compliant contracts that embed the information required for data-subjects to provide informed consent, *(ii)* automatically providing facilities for runtime data flow inspection. We expand on each of these in turn.

Provision of ongoing risk breakdown. Our development environment generates an overall risk rating for apps, based on the aggregate risk of the nodes from which it is composed. Our environment also reflects risks that fall outside the remit of GDPR (such as physical risks mentioned earlier). Each node in the development environment has an in-built schema (provided by the environment, not the developer) that provides, amongst other things, a risk score and breakdown based upon current configuration (e.g., the hardware it works with, the proposed data rate, the particular actuation to be performed). As configuration options are modified and nodes are introduced or removed, the score and breakdown will update to reflect the changes (Fig.3). Those characteristics of a node that will most influence the global risk score are currently *i)* whether it exports any data off the box, *ii)* it triggers physical actuation, *iii)* it utilizes insecure / leaky / non-compliant hardware, *iv)* it uses unverified code or libraries.

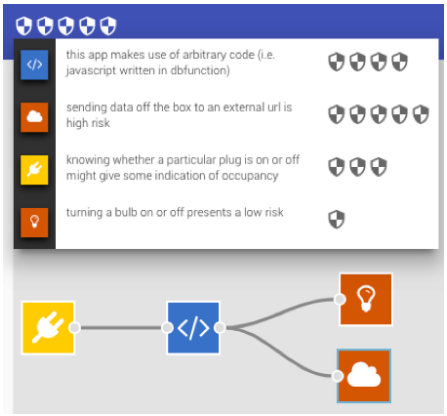

Fig. 3. SDK risk overview

It may be reasonably argued that "risk" is a subjective concept that covers an indeterminate number of possibilities, and will be influenced by more than just an app's construction and configuration (e.g., the deployment environment can profoundly influence risk likelihood and harms). However we would counter that in conceptualising even a crude notion of risk, the environment will sensitize developers to important concerns *in the course of building apps*, i.e. at the point where they are likely to enact change. We view our risk overview as a "placeholder" and expect that further research and subsequent iterations will lead to improved risk calculations.

Our final risk rating and breakdown is also made available to app store users to further motivate and drive the development of low risk and even 'no risk' apps.

Tracking personal data To help developers assess the risks of personal data disclosure, at a minimum, we require they are able to *(i)* differentiate between data that is personal, sensitive or neither and *(ii)* track the flow of personal data, so that processing risk (e.g. inference attacks made possible by combining data) and exposure risk (e.g. location data being exported off the box) can be identified. Our goal is to help developers assess disclosure vulnerabilities prior to deployment (i.e. statically) rather than at run-time; Databox has its own mechanisms for managing dataflows at runtime.

All data that is output from a node has a corresponding personal data schema. The schema allows developers not only to view the flow of personal data through an app, but to reveal points within an app where further personal inference is possible (e.g. when multiple items of personal data or profiling could be combined to infer a new item of personal data). Take, for example, an algorithm that processes a user's gender, postcode and age. These three items may be enough, with minor effort, to infer a user's identity (perhaps using an

Table 2. 6 personal data types

label	type	ordinal	description	example
i1	identifier	primary	data that directly identifies a data subject	full name, picture
i2	identifier	secondary	data that indirectly identifies a data subject	mac address, username, (age, postcode, birthplace)
p1	personal	primary	data that is evidently personal	friends, mortgage, salary
p2	personal	secondary	inferred personal data	gender, age, income (from browsing data)
s1	sensitive	primary	GDPR special categories of data	criminal convictions, health record
s2	sensitive	secondary	inferred sensitive data	race (from postcode), sexuality (from image)

auxiliary public dataset). Less obviously, perhaps: an algorithm that utilises mobile phone accelerometer data may be able, assuming a high enough sampling frequency, to infer a user's height, weight and gender [39] or smart metering data may reveal personal habits [40] or occupancy [41]. As a start, inspired by GDPR, our schema specifies six top-level personal data types (Table 2). In our schema (Table 3), the *(type, ordinal)* attributes establish the top-level type and the *category, subtype and description* attributes (originated by us) provide further context. The schema has a *required* attribute to denote which attributes must be present for a schema to apply. For example, if an IoT camera provides a timestamp, bitmap and light reading, only the bitmap attribute is required for the data to be treated as personal.

Table 3. personal data schema

attribute	description
type	*identifier* | *sensitive* | *personal*
ordinal	*primary* | *secondary*
category	*physical* | *education* | *professional* | *state* | *contact* | *consumption...*
subtype	sensitive will include biometric, health, sexual, criminal. Personal includes education, profession, consumption.
description	details of this particular item of personal data (and method of inference if secondary)
required	list of attributes of this data that must be present in order for this to constitute as personal data

The schema is extended for secondary (i.e. inferred) types, to specify the conditions that must be satisfied to make an inference possible (Table 4). We currently support two types of condition: *(i)* attributes – the additional set of items of personal data items that, when combined could lead to a new inference; *(ii)* granularity – the threshold sampling frequency required to make an inference.

Table 4. personal data schema

attribute	description
confidence	an accuracy score for this particular inference, ranging from 0 to 1
conditions	list of *granularity* | *attribute*
evidence	where possible, a set of links to any evidence that details a particular inference method
status	*inferred* | *inferrable*

When multiple attribute and/or granularity conditions are combined, all must hold for an inference to be satisfied. Finally our status attribute distinguishes between personal data where *(i)* an inference has been made, and *(ii)* the data is available to make inference possible. For example, browsing data and gender may be enough to infer whether an individual is pregnant (i.e. these two items combined make pregnancy inferable) but if a node makes an actual determination on pregnancy, then the resulting data is inferred. We also make two additional assumptions:

- When building a flow, the SDK assumes all data sources belong to the same user. Our next version will formalize this.
- The schema permits data to be tagged as personal even if it is not associated (directly or indirectly) with an individual. Although GDPR specifies that any data that cannot be related to a "natural person" is not personal, we take the view that any items of personal data may still, given the necessary context, be used to identify an individual.

When making use of the schema in the SDK, datasources will define the personal data that they generate, whereas processing and profiling nodes will generate schemas based on the transforms they run on their input data. For example, the combine processing node whose job is to merge attributes from its inputs, auto-generates an output schema by combining the schemas of all input attributes to be merged. Thus it is the SDK's role and not the developer's, to calculate how schemas propagate through an app.

To illustrate a basic example in the SDK, consider Table 5 which outlines the relevant parts of the accelerometer schema for the flows in Fig.4. In the left-hand

Table 5. part of the accelerometer datastore personal schema

attribute	description
type	personal
subtype	gender
ordinal	secondary
required	[x,y,z]
conditions	type: granularity, threshold: 15, unit: Hz

flow, *p2* is output from the accelerometer to show that personal data (i.e. a user's

gender) is *inferable* from the *x,y,z* components of its data (it is semi-transparent to denote it is *inferable* rather than *inferred*). Similarly, with the profile node, i1 is output to show *fullname* is a primary identifier. When these are merged in the combine processor, the output schema will contain the accelerometer's *p2*, and the profile's *i1*. In the right-hand flow, the combine node is configured to only combine the *x* and *y* components of the accelerometer data with the profile data. Since *x,y* and *z* are all marked as required (Table 5) for a gender inference to be possible, the combine node's output schema will only contain *i1* (and not *p2*). The SDK will automatically recalculate and re-represent the flow of personal

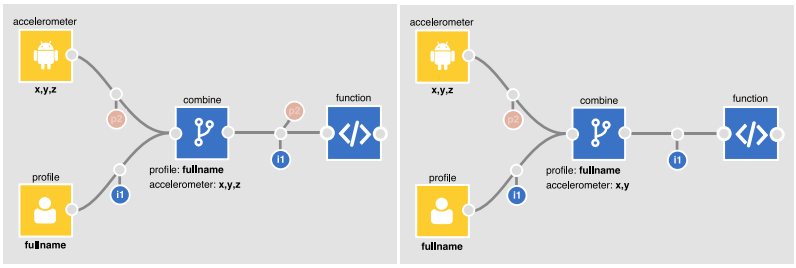

Fig. 4. combining personal data in the SDK

data whenever a node or edge is removed, added or reconfigured. As flows get more complex this becomes invaluable; it helps developers to quickly determine how changes in configuration will alter the flow of personal data.

In tracking personal data the SDK also flags points in an app that may require further attention. When downstream nodes use inferred data with a low confidence score (provided by the schema), developers are warned that processing is based on potentially incorrect data. When any personal data is being exported off the box (i.e. connected to the export node), developers are reminded to ensure data minimisation applies.

Creating GDPR compliant contracts. When a user installs an app on the Databox they are presented with an SLA. The goal of the SLA is to provide transparency and to fulfil the information to be provided to users when personal data are collected (Articles *12-18*). The SLA is a multi-layered notice that furnishes the information in an easily readable format (see [42] for further details). Where appropriate, SLAs enable end-users to exercise granular choice over data sampling and the elements of an app's processing they consent to. SLAs are not static notices then, but dynamic, user-configurable consent mechanisms that surface and articulate who wants to access which connected devices and what they want to process personal data for. They are constructed from a *manifest* file that sets out all possible configurations, and which is submitted alongside an

app when it is published. The SDK streamlines this process; given its knowledge of an app's construction it already knows the data sources being accessed (and at which granularity), the processing taking place and the outputs, all of which are automatically embedded in the manifest. At app publication time, when an app uses multiple data sources, the developer is invited to mark each flow from each source as compulsory or optional, which translates to a set of granular consent options at install time. All that remains is for the developer to provide a description of the app and its benefits, and the remaining statutory information required by GDPR.

Runtime Inspection. Though the development environment ensures that the sources of data that an app operates on and what it outputs to are made transparent, the way in which the app operates, i.e., how a decision is arrived at, or how a data flows through an app remains opaque to a user at app runtime. This becomes an important matter to surface under Article *13* of GDPR, which

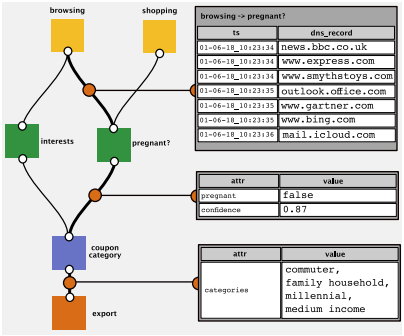

Fig. 5. App inspection interface

requires that meaningful information about the logic involved in automated processing is provided to the data subject. All apps built in the SDK record the path and state of all data as it moves through a flow. SDK apps are all bundled with an interface that uses this path information to make apps 'inspectable' at runtime. By way of example, Fig.5 shows part of an inspection interface on the Databox UI for an app that processes browsing and shopping data to send coupon requests to a third party. The top of the interface shows of the app's datastores, in this case, browsing and shopping. A user can select any node in the path to get a real-time feed of the data entering and exiting it. This is a nascent first step towards satisfying Article *13*. More important at this stage, is that data flow capture is built into apps to support user-inspection interfaces. We are already seeing alternative representations in research [43]; one interesting approach uses 'comic strip' visualisation techniques to communicate the logic of automated processing to end users [44].

4.2 Future Research for the SDK

A number of interesting challenges have emerged which we are keen to explore in greater detail and which are, we think, of broad relevance.

Algorithmic Intelligibility for Developers. Our work on making the operation and intent of apps intelligible to end-users is at an early stage and touches on a rapidly expanding area of research. However research into how an app's processing can be made intelligible to app creators (i.e. developers) is underrepresented in the literature. End-user oriented development environments reduce the competencies necessary for creating apps and expand the cohort of potential app developers. In addition, access to machine learning toolkits such as Google's TensorFlow enable developers to utilise complex machine learning algorithms whilst remaining divorced from all but a rudimentary understanding of the models and logic involved. This makes it increasingly easy for developers to make naïve use of machine-learning algorithms that lead to unfair, incorrect, and ultimately harmful outcomes. Educating and sensitising developers to the implications of the code they create is therefore a worthy goal. As [45] succinctly state: *"in many cases what the data subject wants is not an explanation—but rather for the disclosure, decision or action simply not to have occurred"*.

Articulating risk. Our work on risk assessment in the SDK argues for sensitising developers to the implications of their choices during app construction. Yet, as discussed, our conception of risk is relatively simple. We aim to improve upon this by representing risk as two metrics: likelihood (what is the probability of occurrence?) and harm (what bad things will happen if it does occur?). To make this tractable, the SDK will need to take into account the app's intended deployment context in addition to the personal data it operates on. For example, an app that visualises a user's browsing history on a screen at home will carry different risks from one that exposes the same data to an employer.

5 Conclusion

The emergence of the IoT is driving a shift in data storage and processing to the edge of the network to reduce traffic and latency and to improve resilience and the potential for data privacy. We have argued that GDPR raises an unmet challenge in supporting IoT app development that requires: *(i)* a broad cohort of developers be provided with clear information on the risks that attach to the use of personal data and *(ii)* that all necessary features and information are embedded in apps in order that end-users are provided with the information they need to provide informed consent and the facility to examine an app's operation at runtime. We have presented the design and implementation of a set of developer features (risk breakdown, personal data tracking, compliant contracts and runtime inspection) aimed at meeting these requirements. In doing so, we have taken a step towards identifying how we can improve support for developers who write code to process personal data.

References

1. Fog Computing and the Internet of Things: Extend the Cloud to Where the Things Are, https://www.cisco.com/c/dam/en_us/solutions/trends/iot/docs/computing-overview.pdf. Last accessed 2018/6/26.
2. IDC FutureScape: Worldwide IoT 2018 Predictions, https://www.idc.com/getdoc.jsp?containerId=US43171317. Last accessed 2018/6/26.
3. Amazon Echo, https://en.wikipedia.org/wiki/Amazon_Echo. Last accessed 2018/6/26.
4. Apple HomePod, https://www.apple.com/uk/homepod. Last accessed 2018/6/26.
5. Google Home, https://store.google.com/product/google_home. Last accessed 2018/6/26.
6. Home Assistant, https://www.home-assistant.io. Last accessed 2018/6/26.
7. nCube, https://ncubehome.co.uk. Last accessed 2018/6/26.
8. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation). Official Journal of the European Union, L119:1–88, April 2016.
9. Chaudhry, A., Crowcroft, J., Howard, H., Madhavapeddy, A. ,Mortier, R., Haddadi, H., McAuley, D. : Personal data: thinking inside the box. In Proceedings of the fifth de-cennial Aarhus conference on critical alternatives, pp. 29-32. Aarhus University Press (2015).
10. Lee, S., Wong, E. L., Goel, D., Dahlin, M., Shmatikov, V.: PiBox: A Platform for Privacy-Preserving Apps. In Proceedings of NSDI, pp. 501-514. (2013).
11. Giffin, D. B., Levy, A., Stefan, D., Terei, D., Mazières, D., Mitchell, J. C., Russo, A.: Hails: Protecting Data Privacy in Untrusted Web Applications. In Proceedings of OSDI, pp. 47-60. (2012).
12. Willis, D., Dasgupta, A., Banerjee, S.: ParaDrop: a multi-tenant platform to dynamically install third party services on wireless gateways. In Proceedings of the 9th ACM workshop on Mobility in the evolving internet architecture, pp. 43-48. ACM (2014).
13. Youngblood, G. M., Cook, D. J., Holder, L. B.: A learning architecture for automating the intelligent environment. In Proceedings of the Conference on Innovative Applications of Artificial Intelligence, pp. 1576–1583. MIT Press, Cambridge, MA (2005).
14. How GDPR Will Change the Way You Develop, https://www.smashingmagazine.com/2018/02/gdpr-for-web-developers. Last accessed 2018/6/26.
15. Domoticz, https://domoticz.com. Last accessed 2018/6/26.
16. OpenHAB, https://www.openhab.org. Last accessed 2018/6/26.
17. OpenRemote, http://www.openremote.com. Last accessed 2018/6/26.
18. Project Things, https://iot.mozilla.org. Last accessed 2018/6/26.
19. Cozify, https://en.cozify.fi. Last accessed 2018/6/26.
20. Fibaro, https://www.fibaro.com. Last accessed 2018/6/26.
21. Vera, http://getvera.com. Last accessed 2018/6/26.
22. Mydex, https://mydex.org. Last accessed 2018/6/26.
23. Hub of All Things, https://hubofallthings.com. Last accessed 2018/6/26.

24. Android Things, https://developer.android.com/things/index.html. Last accessed 2018/6/26.
25. Apple HomeKit, https://www.apple.com/uk/ios/home. Last accessed 2018/6/26.
26. Home Assistant, https://www.home-assistant.io. Last accessed 2018/6/26.
27. Samsung SmartThings, http://www.samsung.com/uk/smartthings. Last accessed 2018/6/26.
28. Mark. W. Newman. 2006. Now we're cooking: Recipes for end-user service composition in the digital home. Position Paper– CHI 2006 Workshop IT@Home.
29. IFTTT, https://ifttt.com. Last accessed 2018/6/26.
30. Stringify, https://www.stringify.com. Last accessed 2018/6/26.
31. Yeti, https://getyeti.co. Last accessed 2018/6/26.
32. Zapier, https://zapier.com. Last accessed 2018/6/26.
33. Mi, X., Feng Q., Ying, Z., XiaoFeng, W.: An empirical characterization of IFTTT: eco-system, usage, and performance. In Proceedings of the 2017 Internet Measurement Conference, pp. 398-404. ACM, New York, (2017).
34. Surbatovich, M., Jassim,A., Lujo B., Anupam D., Limin, J.: Some recipes can do more than spoil your appetite: Analyzing the security and privacy risks of ifttt recipes. In Proceedings of the 26th International Conference on World Wide Web, International World Wide Web Conferences Steering Committee, pp. 1501-1510. (2017).
35. Attacking Discrimination in ML. https://research.google.com/bigpicture/attacking-discrimination-in-ml. Last accessed 2018/6/26.
36. Eslami, M., Krishna Kumaran, S.R., Sandvig, C., Karahalios, K.: Communicating Algo-rithmic Process in Online Behavioral Advertising. In Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, p. 432. ACM. (2018).
37. Ribeiro, M.T., Singh, S. and Guestrin, C.: Why should I trust you?: Explaining the predictions of any classifier. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1135-1144. ACM. (2016).
38. Morrison, J.P.: Flow-based programming. In Proceedings of the 1st International Workshop on Software Engineering for Parallel and Distributed Systems, pp. 25-29. (1994).
39. Weiss, Gary M., and Jeffrey W. Lockhart.: Identifying user traits by mining smart phone accelerometer data. In Proceedings of the Fifth International Workshop on Knowledge Discovery from Sensor Data. pp. 61-69. ACM, (2011).
40. Smart meters review TV viewing habits. http://www.h-online.com/security/news/item/Smart-meters-reveal-TV-viewing-habits-1346385.html. Last accessed 2018/6/26.
41. Kim, Y., Schmid, T., Srivastava, M.B., Wang, Y.: Challenges in resource monitoring for residential spaces. In Proceedings of the First ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings, pp. 1-6. ACM, (2009).
42. Crabtree, A., Lodge, T., Colley, J., Greenhalgh, C., Mortier, M.: Building accountability into the Internet of Things: the IoT Databox Model. In Journal of Reliable Intelli-gent Environments. SSRN, (2018).
43. Wang, Q., Hassan, W.U., Bates, A., Gunter, C.: Fear and Logging in the Internet of Things. In Network and Distributed Systems Symposium. (2018).
44. Schreiber, A. and Struminski, R.: Tracing personal data using comics. In International Conference on Universal Access in Human-Computer Interaction. pp. 444-455. (2017).
45. Edwards, L. and Veale, M.: Slave to the Algorithm: Why a Right to an Explanation Is Probably Not the Remedy You Are Looking for, Duke L. and Tech. Rev., 16, p.18. (2017).