Understanding I/O Design Patterns in IoT Applications

Bassma Saleh, Ebelechukwu Nwafor, and Gedare Bloom
Department of Electrical Engineering and Computer Science
Howard University, Washington, DC 20059
Email: bassma.saleh@bison.howard.edu

***Abstract*—The notion of connected heterogeneous devices (things) have witnessed widespread proliferation which can be attributed to some of the benefits it offers such as ease of access through device automation and efficiency of service (e.g smart things). However, security has been a major challenge that has hindered some consumers and major commercial IT Stakeholders from adopting its widespread proliferation. Due to the connection of heterogeneous devices at an unprecedented scale, this opens the door to new attack vectors. Additionally, the limited memory and computation capability of these devices increases the complexity of securing such devices by applying off the shelf, standard security techniques. To model proper threat and attack vectors, it is important to understand common I/O design patterns that exists in various IoT applications. I/O Design Pattern allows for an abstract model representation of data flow semantics which are used by most IoT applications. In this paper, we describe various communication protocols for each communication medium in an IoT ecosystem. we also identify common I/O design patterns for IoT application with an emphasis on data flow in edges devices. This information is a first step to enable modeling and the identification of malicious threats that exists in IoT applications.**

I. Introduction

The Internet of Things (IoT) is transforming home, industrial and commercial automation exponentially and increasing the number of devices connected to the Internet. Cisco estimates that over 50 million devices will be connected to the internet by 2020 [1]. With the increasing amounts of connected heterogeneous devices, security and privacy risks also increase. For example, vulnerabilities in a brand of baby monitors allowed unauthorized access to devices whereby a malicious intruder can view live feeds from a remote location [2].

Due to the heterogeneity of devices and and the sensitivity of data generated on IoT devices, trust is a critical step to ensuring the security of IoT devices. This can be achieved through data provenance which is a comprehensive history of activities that occurred on an entity from its origin to its current state. Provenance ensures confidence in the fidelity of data. Provenance has been applied in domains such as scientific workflows for experiment reproducibility, information security as a form of access control, and also for intrusion detection. For IoT devices (things) that produce lots of sensor-actuator data, a workflow representation of sensor data can depict dependency between sensor readings and information stored or transmitted by the device.

In this paper, we introduce Provenance Aware Internet of Things System (PAIoTS), a provenance collection framework for IoT devices, in which provenance data is collected and modeled to represent dependencies between sensor-actuator readings and IoT entities. Most of the interconnected heterogeneous devices are embedded systems that require lightweight and resource-efficient solutions as compared to general purpose systems. This requirement is attributed to the constrained memory and computing power of such devices. We also contribute a provenance sensor model which provides a means to convert sensor event traces to provenance model graphs.

II. Background

In this section, we describe key concepts of data provenance, IoT characteristics, and provenance models. We also provide a smart home example for provenance collection in the IoT.

A. *Internet of Things*

In 1991, Mark Weiser, a pioneer of ubiquitous computing envisioned the notion of computing interleaved in our daily lives [3]. Kevin Ashton, an early pioneer of IoT, defines the IoT as devices in our everyday lives which can be identified connected to a network [4]. These devices can learn from information gathered autonomously without human input which allows for improvements in waste reduction and overall standard of living. Buckley et al. [5] define the IoT as a network of billions of machines communicating with each other. Gubbi et al [6] defines the IoT as an interconnection of sensing and actuating devices that allows data sharing across platforms through a centralized framework.

We define the IoT as a network of heterogeneous devices with sensing and actuating capabilities connected to internet-based cloud services. IoT has applications in home automation, smart health, automotive communication, machine to machine communication, industrial automation. Pervasive connectivity between heterogeneous devices allows them to share information with each other and with cloud based data analytics, which drives IoT. With analytics, IoT applications can learn from user data to make smarter decisions.

IoT architecture represents a functional hierarchy of how information is disseminated across layers between devices which contain sensing and actuating capabilities and massive data centers (cloud storage). Figure 1 displays the IoT architecture and the interactions between the respective layers:

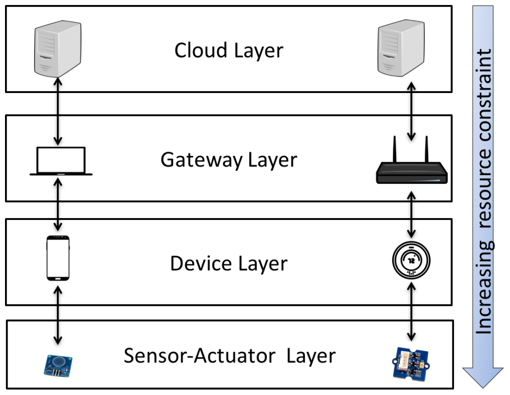

Fig. 1. IoT Architecture Diagram. The arrows illustrates the interaction between data at various layers on the architecture.

sensor and actuator, device, gateway and cloud. The base of the architectural stack consists of sensors and actuators that gathers information from the physical world (via sensors) and manipulates it (via actuators) while interacting with the device layer. The device layer aggregates data from sensors and actuators and forwards data to the gateway layer. The gateway layer routes and forwards this data collected from the device layer to the cloud layer for storage and processing. Resource constraints decrease up the architectural stack, with the cloud layer having the most resources (memory, power, computation), and the sensor-actuator layer having the least.

B. Motivating Use Case

Creating a provenance collection system is beneficial to IoT because it provides a means of verifying the integrity of data in the heterogeneous interconnected devices thereby building trust. Enabling provenance collection in IoT devices allows devices to capture valuable information which enables backtracking in an event of a malicious attack.

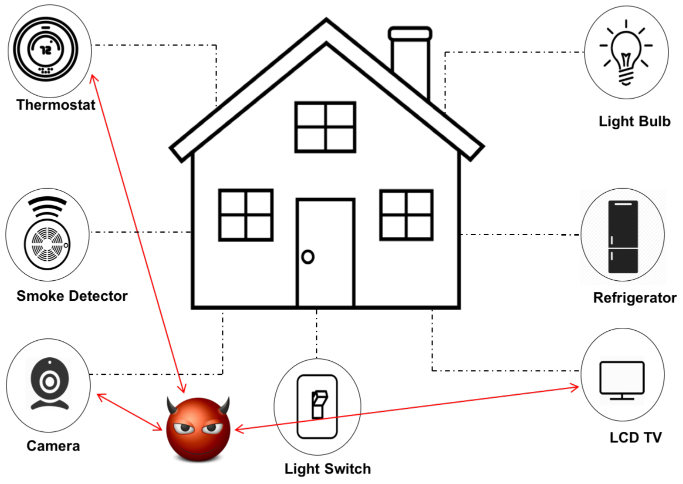

Fig. 2. Smart home use case scenario which demonstrates how provenance can be used to detect a point of entry of malicious intrusion

Consider a smart home as illustrated in Figure 2 which contains interconnected devices such as a thermostat which automatically detects and regulates the temperature of a room based on prior information of a user's temperature preferences, a smart lock system which can be controlled remotely and informs a user via short messaging when a door has been opened, a home security camera monitoring system, a smart fridge which sends a reminder when food products are low etc. In an event that a malicious intruder attempts to gain access to the security camera remotely, provenance information can be used to track the series of events to determine where and how a malicious attack originated. Provenance can also be used as a safeguard to alert of a possible compromise thereby protecting against future or ongoing attacks.

C. Data Provenance

The Oxford English dictionary defines provenance as "the place of origin or earliest known history of something" [7]. An example of provenance can be seen with a college transcript. A transcript is the provenance of a college degree, because it identifies all of the courses satisfied in order to attain the degree. In the field of computing, data provenance, also known as data lineage, is the history of all activities performed on an entity from its creation to its current state. Cheney et al. [8] describe provenance as the origin and history of data from its lifecycle. Buneman et al. [9] describe provenance from a database perspective as the origin of data and the steps in which it is derived in the database. We define data provenance of an entity as a comprehensive history of activities that occur on that entity from its creation to its present state.

Provenance is easily represented as an acyclic graph which denotes causal relationships and dependencies between entities. Provenance consists of the following characteristics:

- Who: Provides information linking activities to an entity. Knowing the "who" characteristic is essential because it maps the identity of modification to a particular data object. An example of "who" in an IoT use case is a sensor device identifier.
- Where: Location information at which data transformation was made. This provenance characteristic could be optional since not every data modification contains location details. An example is a GPS coordinate.
- When: The time at which data transformation occurred. This is an essential provenance characteristic. Being able to tell the time of a data transformation allows for tracing data irregularities. An example of this characteristic is a timestamp that denotes when sensor data was read.
- What: The transformation applied on operations (create, read, update, and delete) that can be performed on a IoT data object.

D. Model for Representing Provenance for IoT

Provenance of sensor readings in an IoT device should describe the dependency relationships between all entities responsible for producing those readings. We adopt the Provenance Data Model (PROV-DM) [10], a W3C standard which conforms to Provenance Ontology (PROV-O) and is used to depict dependencies between entities, activities and agents (digital or physical). PROV-DM creates a common model that allows for interchange of provenance information between heterogeneous devices and is represented serialized in three formats: XML, JSON and RDF.

PROV-DM contains two major components: types and relations. Types can be entities, activities, or agents. An entity is a physical or digital object. An activity represents some form of action that occurs over time. An agent takes ownership of an entity, or performs an activity. Figure 3 illustrates the types and relations contained in PROV-DM and their graphical representation. Entities, activities and agents are represented as oval, rectangular and pentagonal shapes respectively.

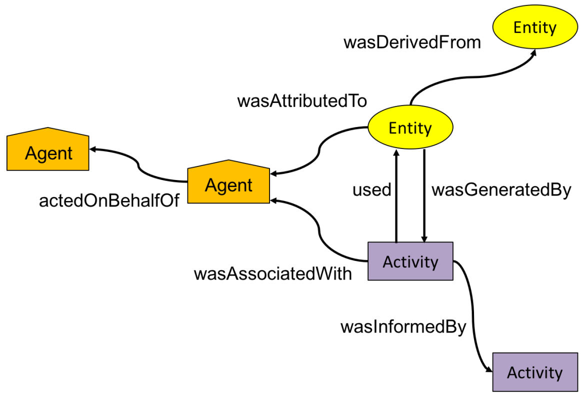

Fig. 3. Prov-DM respresentation showing types in the model (Entity, Activity, and Agent) and the relationships between them

PROV-DM defines the following seven relationships between the types.

- wasGeneratedBy: Signifies the production of a new entity by an activity.
- used: An entity generated by one activity has been adopted by another activity.
- wasInformedBy: Signifies the exchange of an entity by two activities.
- wasDerivedFrom: Represents a copy of information from an entity.
- wasAttributedTo: Denotes relational dependency between an entity and an agent when the activity that created the agent is unknown.
- wasAssociatedWith: An agent created or modified the activity
- actedOnBehalfOf: Delegation of authority from an agent to itself or another agent to perform a particular responsibility.

III. Provenance-Sensor Model

In this section, we introduce the Provenance-Sensor Model and explain how PROV-O is used to convert sensor readings to provenance. Sensor data contains observation information such as temperature, and location details which can be transformed to standardized data interchange formats (RDF, XML, JSON). Sensor data are time series data which can be traced over time. Trace data containing sensor readings are important but do not depict dependency relationships when used alone. We transform trace data to provenance to represent causality and dependency relationships between entities in an IoT system. Provenance can be represented as a directed acyclic graph and the edge between two entities is considered a relation. Relations between data objects follows provenance ontology which depicts transformation between entities. We integrate PROV-O and sensor data for better representation of dependency relationships between trace data generated.

A single sensor might posses the ability to collect multiple trace data, td. For example, a sensor might be able to collect sensor readings of temperature, location, humidity. A combination of trace data at a particular point in time is considered an event. We define an event for sensor s_1 at time t as $e = \{td_1, ...td_n\}$ where td_1 is the first trace data collected by s_1 and td_n is the last which occurs at time t.

Adopting provenance ontology to IoT, we represent device information as agents (prov:agents), the operation performed on sensor readings (read, create, update) as a provenance activity (prov:activity), and events as entities (prov:entity). A sensor trace is defined as a tuple (t, e, a, s_1, r_1) where t represents a timestamp, e an event, a an operation, s_1 sensor information and r_1 device information.

1) Device with one sensor: Consider a humidity and temperature sensor s_1, connected to device r_1, a Raspberry Pi. Event $e = \{temperature, humidity\}$ therefore the tuple representation of trace data for sensor s_1 at time t_1 is $(t_1, \{temperature, humidity\}, a, s_1, r_1)$. a is the operation performed on the sensor. Each tuple is mapped to the Provenance Ontology representation using the defined constructs in section IV. Since s_1 is contained in device r_1, r_1 forms an edge with s_1 with the used relation. (e.g r_1 used s_1).

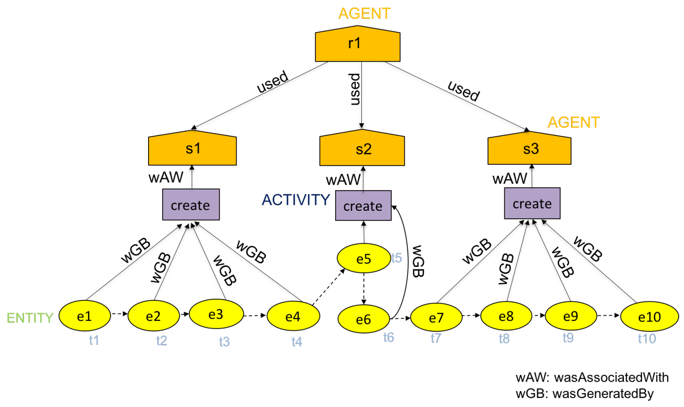

Fig. 4. Provenance-Sensor Model

2) Device with multiple sensors: Figure 4 further illustrates the concept of mapping provenance to sensor data using graphical representations from PROV-DM relations and types. The figure depicts a device, r_1, connected to three sensors s_1, s_2, and s_3 with events $[e_1, e_2, ..., e_n]$. Data is generated by three identical temperature sensors, s_1, s_2 and s_3. The graph represents data dependency between r_1, the three sensors, the activity performed by the sensors (the sensors generate data in this case) and the events. For each sensor, the total number of tuple is equal to the number of events. For example, sensor trace data from sensor s_1 is represented by four tuples: $(t_1, e_1, create, s_1, r_1)$,$(t_1, e_2, create, s_1, r_1)$, $(t_1, e_3, create, s_1, r_1)$, and $(t_1, e_4, create, s_1, r_1)$. Each event makes an edge with the preceding event. Edges between events are denoted with a dotted arrow. This represents time dependency between events and that each event occurs consecutively at distinct time intervals.

Algorithm 1 presents the steps taken by PAIoT to map sensor trace data into graph based provenance. F represents a list of k tuples. For each tuple contained in F, s, and r_1 represents sensor and device information respectively and are defined as agents. e is defined as an event, a is defined as an activity. p is a memory representation of provenance information containing all provenance types and their relations. x and y are a list of relations between sensor-device and activity-sensor, respectively.

Algorithm 1: Provenance-Sensor Mapping

```
Function trace2Prov (F)
    p ← createProvDocument()
    for  $k \in F$ do
        if  $s \notin p$ then
            s ← createAgent()
        if  $r_1 \notin p$ then
            $r_1$ ← createAgent()
        e ← createEntity()
        a ← createActivity()
        if  $x \notin p$ then
            x ← relateSensorToDevice()
        if  $y \notin p$ then
            y ← relateActivityToSensor()
        z ← relateEntityToActivity()
    return p
```

IV. PAIoTS System Implementation

In this section, we outline PAIoTS, a trace-based provenance collection system for IoT devices. Figure 5 displays the system architecture. Sensor readings in the form of input and output (I/O) events are recorded by the tracer component. This component intercepts I/O and produces trace information represented in Common Trace Format (CTF).

PAIoTS converts CTF trace data to provenance in our Provenance-Sensor Model. This conversion can happen at any layer of the IoT stack. CTF conversion to PROV-DM is done using babeltrace. Babeltrace is a plugin framework which allows the conversion of CTF traces into other formats. Trace or provenance data is securely transmitted to a gateway and later transmitted and stored in a cloud backend. Our backend of choice is Neo4j, a graph database with support for efficient storage, query and visualization of provenance data.

CTF contains a mandatory stream known as metadata. Metadata contains information about other streams. It allows parsing a stream without specifying a layout. CTF encodes binary trace output information containing multiple streams of binary events such as I/O activity. Each event can be broken into streams. Streams allow for fast processing since they do not have to be stored in disk before being sent over a network or processed in memory.

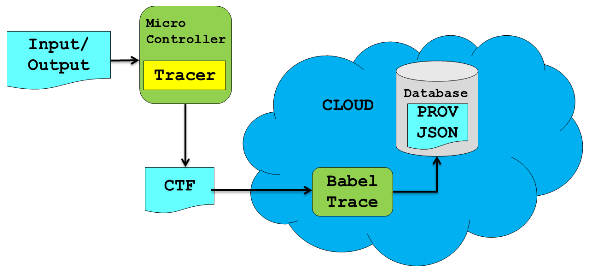

Fig. 5. System Architecture for PAIoTS.

Our provenance collection system records transformations of I/O data for devices connected in the IoT. For our implementation, we use several tools and hardware components in the development of our prototype:

- Raspberry Pi is the microcontroller used to demonstrate our approach. We chose Raspberry Pi because it is a representation of what can be found on an IoT gateway device. Raspberry Pi is a low cost, simple IoT demonstrator.
- Neo4j is a graph database which allows optimized querying of graph data such as provenance.
- Babeltrace is a trace converter tool to convert traces from one format into another.
- barectf is a light weight generator of C code that generates trace data in CTF.
- A yaml generator in barectf creates yaml configuration files with information babeltrace needs to generate CTF trace output. Configuration files contain settings such as an application trace stream, packet type, payload type and size.

V. Related Work

Muniswamy-Reddy et al. [11] developed Provenance Aware Storage System (PASS), a provenance collection system that tracks system-level provenance of Linux file system. Provenance information is stored in the same location as the file system for easy accessibility, backup, restoration, and data management. Provenance information is collected and stored in the kernel space. PASS is composed of 3 major components: provenance collector, provenance storage, and provenance query. The collector keeps track of system level provenance. It intercepts system calls which are translated into provenance data and initially stored in memory. Provenance data is then transferred to a file system in a kernel database, BerkleyDB which maps key value pairs for provenance data for fast index look up. Our approach addresses application level provenance for memory constrained embedded systems.

Bates et al. [12] developed, HiFi, a system level provenance collection framework for the Linux kernel using Linux Provenance Modules (LPM), which utilizes Linux Security Modules (LSM). LSM is a framework that was designed for providing custom access control inside the Linux kernel. HiFi contains three components: provenance collector, provenance log and provenance handler. The collector and log are contained in the kernel space while the handler is contained in the user space. The collector uses LSM to record provenance data and writes it to the provenance log. The handler reads the provenance record from the log. The log is a storage medium which transmits the provenance data to the user space. This approach to collecting provenance data differs from our work since we focus on memory constrained embedded systems which might not contain an operating system or a file system.

RecProv [13] is a provenance system which records user-level provenance, avoiding the overhead incurred by kernel level provenance recording. It does not require changes to the kernel like most provenance monitoring systems. It uses Mozilla rr to perform deterministic record and replay by monitoring system calls and non deterministic input. The provenance information generated is converted into PROV-JSON, and stored in Neo4j, a graph database for visualization and storage of provenance graphs. In PAIoT, we convert trace data to provenance and also use Neo4j for storage and visualization of the provenance data however, our approach focuses on the relationship between entities in a device with limited computation and memory capabilities and also the transformation of sensor data in these devices.

Spillance et al. [14] developed a user space provenance collection system, Storybook that allows the use of application specific extensions such as database provenance, system level provenance, web and email servers. Storybook captures provenance by intercepting system level events in the FUSE file system and stores provenance data in MySQL. StoryBook allows developers to implement provenance inspectors custom provenance models for specific applications which are often modified by different application (e.g web servers, databases). When an operation is performed on a data object, the appropriate provenance model is triggered and provenance data for that data object is captured. StoryBook stores provenance information such as open, close, read or write, application specific provenance, and causality relationship between entities contained in the provenance system. Provenance data is stored in key value pairs using Stasis and Berkely DB. In our approach to provenance collection, we are particularly interested in the provenance of sensor data in memory constrained embedded systems.

Lim et al. [15] developed a model for calculating the trust of nodes in a sensor network by using data provenance and data similarity as deciding factors to calculate trust. The value of provenance signifies that the more similar a data value is, the higher the trust score. Also, the more the provenance of similar values differ, the higher their trust score. The trust score of a system is influenced by the trust score of the sensor that forwards data to the system. Provenance is determined by the path data travels through the sensor network. This work differs from our approach since the authors focus on creating a trust score and do not emphasize how the provenance data is collected.

Compton et al. [16] defines a model for the alignment of Semantic Sensor Network (SSN), a semantic ontology for representing sensor observation data and PROV-O. This model contains details on how the sub-components of SSN can be represented as provenance ontology. The model is only suited only SSN and does not address other sensor semantic representations. Our work focuses on providing a general Provenance-Sensor alignment which is not tied to a specific semantic ontology.

VI. Conclusion and Future Work

In this paper, we motivate the need for integrating provenance into the IoT system. We propose PAIoTS, a provenance collection framework that provides provenance collection capabilities for devices in an IoT system. We plan to evaluate PAIoT with IoT specific performance benchmarks.

Acknowledgment

This research has been supported in part by US National Science Foundation (CNS grant No. 1646317), and by Leidos. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of NSF or Leidos.

References

[1] D. Evans, "The internet of things how the next evolution of the internet is changing everything," https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf, Apr 2011, accessed: 2016-10-01.

[2] M. Stanislav, "Hacking iot: A case study on baby monitor exposures and vulnerabilities," https://www.rapid7.com/docs/Hacking-IoT-A-Case-Study-on-Baby-Monitor-Exposures-and-Vulnerabilities.pdf, Sep 2015, accessed: 2016-10-01.

[3] M. Weiser, "The computer for the 21st century," *Scientific american*, vol. 265, no. 3, pp. 94–104, 1991.

[4] K. Ashton, "That internet of things thing," *RFiD Journal*, vol. 22, no. 7, pp. 97–114, 2009.

[5] L. Yan, Y. Zhang, L. T. Yang, and H. Ning, *The Internet of things: from RFID to the next-generation pervasive networked systems*. CRC Press, 2008.

[6] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, Sep. 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X13000241

[7] Rationality., *The Cambridge Dictionary of Philosophy*. Cambridge University Press, 1999.

[8] J. Cheney, L. Chiticariu, and W.-C. Tan, "Provenance in Databases: Why, How, and Where," *Found. Trends databases*, vol. 1, no. 4, pp. 379–474, Apr. 2009. [Online]. Available: http://dx.doi.org/10.1561/1900000006

[9] P. Buneman, S. Khanna, and T. Wang-Chiew, "Why and Where: A Characterization of Data Provenance," in *Database Theory ICDT 2001*, ser. Lecture Notes in Computer Science, J. V. d. Bussche and V. Vianu, Eds. Springer Berlin Heidelberg, Jan. 2001, no. 1973, pp. 316–330, dOI: 10.1007/3-540-44503-X_20. [Online]. Available: http://link.springer.com/chapter/10.1007/3-540-44503-X_20

[10] "PROV-DM: The PROV Data Model," https://www.w3.org/TR/prov-dm/, accessed: 2016-10-01. [Online]. Available: https://www.w3.org/TR/prov-dm/

[11] K.-K. Muniswamy-Reddy, D. A. Holland, U. Braun, and M. Seltzer, "Provenance-aware Storage Systems," in *Proceedings of the Annual Conference on USENIX '06 Annual Technical Conference*, ser. ATEC '06. Berkeley, CA, USA: USENIX Association, 2006, pp. 4–4. [Online]. Available: http://dl.acm.org/citation.cfm?id=1267359.1267363

[12] D. J. Pohly, S. McLaughlin, P. McDaniel, and K. Butler, "Hi-Fi: Collecting High-fidelity Whole-system Provenance," in *Proceedings of the 28th Annual Computer Security Applications Conference*, ser. ACSAC '12. New York, NY, USA: ACM, 2012, pp. 259–268. [Online]. Available: http://doi.acm.org/10.1145/2420950.2420989

[13] Y. Ji, S. Lee, and W. Lee, "RecProv: Towards Provenance-Aware User Space Record and Replay," in *Provenance and Annotation of Data and Processes*, ser. Lecture Notes in Computer Science, M. Mattoso and B. Glavic, Eds. Springer International Publishing, Jun. 2016, no. 9672, pp. 3–15, dOI: 10.1007/978-3-319-40593-3_1. [Online]. Available: http://link.springer.com/chapter/10.1007/978-3-319-40593-3_1

[14] R. Spillane, R. Sears, C. Yalamanchili, S. Gaikwad, M. Chinni, and E. Zadok, "Story Book: An Efficient Extensible Provenance Framework," in *First Workshop on the Theory and Practice of Provenance*, ser. TAPP'09. Berkeley, CA, USA: USENIX Association, 2009, pp. 11:1–11:10. [Online]. Available: http://dl.acm.org/citation.cfm?id=1525932.1525943

[15] H.-S. Lim, Y.-S. Moon, and E. Bertino, "Provenance-based Trustworthiness Assessment in Sensor Networks," in *Proceedings of the Seventh International Workshop on Data Management for Sensor Networks*, ser. DMSN '10. New York, NY, USA: ACM, 2010, pp. 2–7. [Online]. Available: http://doi.acm.org/10.1145/1858158.1858162

[16] M. Compton, D. Corsar, and K. Taylor, "Sensor data provenance: Ssno and prov-o together at last." 2014.