SECURE DATA PROVENANCE FOR INTERNET OF THINGS(IOT) DEVICES

EBELECHUKWU NWAFOR

Department of Computer Science

APPROVED:

Gedare Bloom, Ph.D.

Legand Burge, Ph.D.

Wayne Patterson, Ph.D.

Charles Kim, Ph.D.

Gary L. Harris, Ph.D.
Dean of the Graduate School

SECURE DATA PROVENANCE FOR INTERNET OF THINGS(IOT) DEVICES

by

EBELECHUKWU NWAFOR, M.S.

DISSERTATION PROPOSAL

Presented to the Faculty of the Graduate School of

Howard University

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

Department of Computer Science

HOWARD UNIVERSITY

FEB 2016

Acknowledgements

First and foremost, I would like to thank God for making this possible, without him i will not be where i am today. I would also like to thank my parents, Benneth and Chinwe for their constant encouragement, love and support.

I would like to thank my advisor Dr. Gedare Bloom for his guidance and encouragement. Also for believing in me and my research ideas. I would like to thank my lab partner Habbeeb Olufowobi and fellow graduate student Marlon Mejias for their constant constructive criticism and valuable input in various drafts revisions of my proposal.

Abstract

The concept of Internet of Things (IoT) offers immense benefits by enabling devices to leverage networked resources thereby making more intelligent decisions. The numerous heterogeneous connected devices that exist throughout the IoT system creates new security and privacy concerns. Some of these concerns can be overcome through data trust, transparency, and integrity, which can be achieved with data provenance. Data provenance, also known as data lineage, provides a history of transactions that occurs on a data object from the time it was created to its current state. Data provenance has been explored in the areas of scientific computing and business to track the workflow of processes,as well as in field of computer security for forensic analysis and intrusion detection. Data provenance has immense benefits in detecting and mitigating current and future malicious attacks.

In this dissertation proposal, we explore the integration of provenance within the IoT architecture. We take an in-depth look at the creation, security and applications of provenance data in mitigating malicious attacks. We define our model constructs for representing IoT provenance data. We propose a provenance aware system that collects provenance data in an IoT framework. We focus on a holistic approach in which provenance information is represented at all layers of the IoT architecture. This system ensures trust and helps establish causality for decisions and actions taken by an IoT connected system. However, there are existing issues with devices running memory space, and these present as a consequence of the amount of data generated by our data provenance collection system, as well as resource constrains on low memory embedded systems. To address these issues, we propose a framework that provides an efficient storage of provenance data contained in an IoT architecture. We evaluate the effectiveness of our proposed frameworks by looking at the applications of provenance data for the security of malicious attacks. We focus on evaluation using an Intrusion Detection System, which detects malicious threats that exists in an IoT framework.

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

In recent years,the Internet of Things(IoT) has gathered significant traction which has led to the exponential increase in the number of devices connected to the internet. According to a report released by Cisco, it is estimated that a total of 50 million devices will be connected to the internet by the year 2020. Data is generated in an enormous amount in real-time by sensors and actuators contained in these devices. With the vast amounts of connected heterogeneous devices, security and privacy risks are exponentially increased. Rapid7, an internet security and analytics organization, released a report highlighting vulnerabilities that exists on select IoT devices. In their report, they outline vulnerabilities in baby monitors which allowed intruders unauthorized access to devices whereby a malicious intruder can view live feeds from a remote location. Having a provenance aware device will be of immense benefit to the security of the device since it ensures trust between interconnected devices. With Provenance information, we can generate an activity trail which can be further analyzed to determine the who, where, and how, in the event of a malicious activity.

In an IoT system, most of the interconnected heterogeneous devices (things) are embedded systems which require lightweight and efficient solutions as compared to general purpose systems.This requirement is attributed to the constrained memory and computing power of such devices. A major issue arise in ensuring that data is properly secured and disseminated across an IoT architecture. The vast amount of data generated from IoT devices requires stronger levels of trust which can be achieved through data provenance. Prove-

nance is used to determine causality and effect of operations performed on data objects. Data provenance ensures authenticity, integrity and transparency between information disseminated across an IoT system.This level of transparency can be translated to various levels across the IoT architectural stack.To achieve transparency in an IoT framework, it is imperative that data provenance and IoT systems be combined to create a provenance aware system that provides detailed records of all data transactions performed on devices connected in an IoT network.

1.2 Provenance-Aware IoT Device Use Case

Consider a smart home which contains interconnected devices such as a thermostat connected a wireless network which automatically detects and regulates the temperature of a room based on prior information of a user's temperature settings, a smart lock system which can be controlled remotely and informs a user via short messaging when a breach has occurred, a home security camera motoring system, A smart fridge which sends a reminder when food produce are low. A malicious intruder successfully attempts to gain access to the smart lock system and security camera remotely. Provenance can be used to track the series of events to determine where and how a malicious attack originated.It can also be used as a safeguard to alert of a possible remote or insider compromise thereby protecting future or current malicious attacks.

1.3 Research Questions

Data provenance and IoT are two important components that need to be unified. However, data provenance collection in IoT raises some important research issues. Some of which are as a result of preexisting provenance and IoT issues. Some of the issues raised as a result of the unification of data provenance with IoT are outlined below:

- Memory constraints on IoT Devices: The vast amounts of data generated leads high

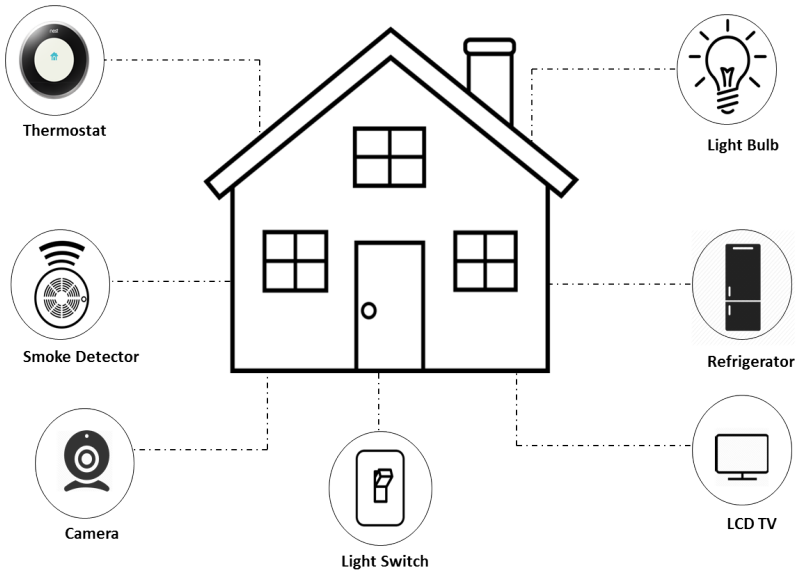

Figure 1.1: Smart home use case Diagram

storage space utilization . Proper memory management/data pruning techniques should be employed for efficient storage of provenance data on memory constrained devices.

- A fundamental question exists in effective modeling of provenance data. How do we model provenance data collected for IoT devices? Are there models used to represent causality between sensor and actuator readings in the IoT architecture.

- How do we effectively collect provenance data in resource constrained device and relate this information accross various layer of the IoT architecture

1.4 Research Contribution

In this dissertation, we propose the following key contributions:

- A provenance collection framework which denotes causality and dependencies between entities contained in an IoT system.This system creates the groundwork for capturing

and storing provenance data on IoT devices.

- An efficient algorithm for storing provenance data on IoT devices. We evaluate our contributions for an efficient provenance storage by using an intrusion detection system on IoT devices.

1.5 Organization of Dissertation proposal

The remaining portion of the dissertation proposal is organized as follows. Chapter 2 talks about background information on data provenance, some of the techniques involved in collecting system level provenance, data pruning techniques for storage optimization and also applications of data provenance with provenance based intrusion detection system. chapter 3 discusses our proposed provenance collection system and focuses specifically on preliminary work done in creating a provenance aware system. Chapter 4 concludes the proposal and discusses the proposed framework and projected timeline for completion.

Chapter 2

Background Information

This chapter describes key concepts of data provenance, IoT characteristics, and provenance models. It outlines differences that exists between provenance and log also provenance and metadata.

2.1 Data Provenance

The oxford English dictionary defines provenance as the place of origin or earliest known history of something. Provenance was associated with art since most art work determine the origin of an art work. An example of provenance can be seen with a college transcript. A transcript can be defined as the provenance of a college degree because it outlines all of the courses satisfied in order to attain the degree.

In the field of computing, data provenance also known as data lineage can be defined as the history of all transformations performed on a data object from the its creation to its current state. Cheney et al decribes provenance as the origin and history of data from its lifecycle. Buneman et al describes provenance from a database perspective as the origin of data and the steps in which it is derived in the database system. We formally define provenance as follows:

Definition 1 *Data provenance of an object can be defined as a comprehensive history of transformations that occurs on that object from its creation to its present state.*

There are two types of provenance. Fine-grained provenance which involves the collection of a whole system detailed provenance data. Coarse grained provenance on the other hand

involves provenance collection of a summarized version of provenance data.Data provenance has immense applications and has been explored in the areas of scientific computing to track how an experiments are produced, in business to determine the work-flow of a process, in field of computer security for forensic analysis and intrusion detection. Provenance denotes the who, where and why of data. An example of provenance for a software systems is a web server's log file. This file contains metadata for various request and response time and ip addresses of all host systems that requests information from the server.Provenance data is represented as an acyclic graph which denotes casual relationship and dependencies between entities. Provenance ensures trust and integrity of data. The information in which provenance offers can be used in digital forensics to investigate the cause of a malicious attack and also in intrusion detection systems to further enhance the security of computing devices.

2.1.1 Provenance Characteristics

provenance denotes the who, where and why of data transformation. In an IoT architecture, It is imperative that we collect sensor and actuator reading that satisfies the required conditions. The characteristics of data provenance are outlined in details below.

- Who: This characteristics provides information on who made modifications to a data object. Knowing the "who" characteristics made is essential because it maps identity of modification to a particular data object. An example of who in an IoT use case is a sensor device ID.

- Where: This characteristic denotes location information in which data transformation was made. This characteristic is optional since not every data modification contains location details.

- When: This characteristic denotes the time information in which data transformation occurred. This is an essential provenance characteristic. Being able to tell the time of a data transformation allows for varied use scenarios of tracing data anomalies.

- How: This characteristic denotes the process in which transformation is applied on a data object. A use case for IoT can be seen in the various operations(delete, create, open) that could be performed on a file.

2.2 Internet of Things(IoT)

There is no standard definition for IoT, however, researchers have tried to define the concept of connected "things". The concept of IoT was proposed by Mark Weiser in the early 1990 which represents a way in which physical objects can be linked to the digital world. Gubbi et al defines the Internet of Things as an interconnection of sensing and actuating devices that allows for the sharing of data across platforms through a common platform.We define the internet of things(IoT) as the following:

Definition 2 *An IoT framework can be defined as a network of heterogeneous devices with sensing and actuating capabilities communicating with each other thereby enabling each device become smarter through data analytics.*

The notion of internet of things has been attributed to smart devices. It is believed that the interconnectivity between various heterogeneous devices allows for devices to share information in a unique and new way. Analytics is the driving force for IoT. With analytics, we are able to learn user patterns and this information in turn can be used to make smarter device decisions. This notion of smart devices can be seen in various commercial applications such as smartwatches to smart thermostats that automatically learns a user pattern and regulates the temperature based on the data collected. The ubiquitous nature of these devices make them ideal choices to be included in consumer products.

With the recent data explosion due to the large influx in amounts of interconnected devices due to IoT, information is disseminated at vast rate and with this increase involves security and privacy issues. Creating a provenance aware system is beneficial to IoT because it ensures the trust and integrity of device. Enabling IoT device provenance aware allows

devices to capture information such as the who, where and how transformations occur on a data object which enables backtracking in an event of a malicious attack.We take a holistic approach to provenance collection by looking at how provenance information is collected accress the IoT architectural framework from bottom(sensor layer) to the top(cloud layer).

2.2.1 IoT Architecture

An IoT architecture represents a hierarchical view of how information is disseminated across multiple devices contained from little sensors to massive data centers(cloud storage). It consists of four distinct layers: The sensor layer, device layer, gateway layer and the cloud layer. The base of the architectural stack consist of sensors and actuators which collect information from IoT devices and interacts with the device layer. The device layer consists of devices(e.g mobile phones, laptops, smart devices) which are responsible for aggregating data collected from sensors and actuators. These device in turn forwards the aggregated data to the gateway layer.The gateway layer is concerned with the routing and forwarding of data collected. It could also serve as a medium of temporal storage and processing of information. This layer serves as the intermediary between the device and the cloud layer. The cloud layer is involved with the storage and processing of data collected from the gateway layer. This information can be further analyzed with advanced analytics to derive insights from the data received. Figure 2 displays the IoT architecture and the interactions between the respective layers. The resource constraints(memory , computation) decreases as you go up the architectural stack with the cloud layer having the most resource and the sensor layer having the least amount of resources

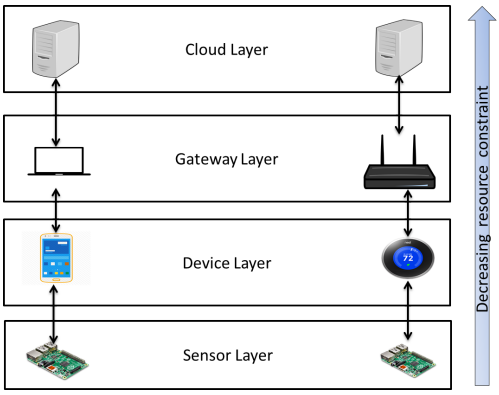

Figure 2.1: IoT Architecture Diagram

2.3 Comparison of Provenance with Log data and metadata

Provenance data, log data and metadata are key data concepts that often are used interchangeably. They are difference and similarities that exists between these concepts. We try to address the difference and similarities that might exist between provenance data, metadata and Log data.

2.3.1 Provenance and Metadata

Metadata and provenance are often considered related but yet subtle exists. Metadata contains descriptive information about data. Metadata can be considered as provenance when there exists relationship between objects and explains how objects are . For example a web server can have metadata information such as the host and destination IP addresses which might not be considered as provenance information for a specific application.It could also contain timestamps with location of IP addresses which can be considered as provenance information. In summary metadata and provenance are not completely not the same, however an overlap exists. Metadata contains valuable provenance information but not all

provenance information is metadata.

2.3.2 Provenance and Log data

Log data contains information about the activity of an object in a operating system or process.Provenance is a specialized form of Log data. It contains information specific to an application domain. Log files might contain unrelated information such as error messages, warnings which might not be considered as provenance data. Logging is limited, lacking information that shows the internal state of an object. Provenance allows for a specified collection of information that relates to the change the internal state of an object.

2.4 Model for representing provenance for IoT

In order to ensure that we properly represent the right kind of provenance information, we need to satisfy the who, where, how, and what of data transformations. Provenance data is represented using a provenance model, PROVDM which is represented in serialized form as JSON object. This model displays the causal relationship of all of the data objects contained in an IoT architectural stack. In this model is contained information such as sensor readings, device name, and device information. This information is mapped into an appropriate provenance data model to allow for interoperability and visualization. There are two models for representing provenance. These models have been applied in various literature and is considered a standard for representing provenance. Details on the provenance models are outlined below:

2.4.1 Open Provenance Model(OPM)

Open Provenance Model is a specification that was derived as a result of a meeting at the International Provenance and Annotation Workshop (IPAW) workshop in May 2006. OPM was created to address the need of allowing a unified model of representing provenance data

amongst various applications. It allows for interoperability between various provenance models that might exist with various applications. The goal of OPM is to develop a digital representation of provenance for entities regardless of if it is produced by a computer system or not. OPM is represented as a directed acyclic graph which denotes causal dependency between objects. The edges in the graph denotes dependencies with its source denoting effect and its destination denoting cause. The definitions of the edges and their relationships are denoted below:

- wasGeneratedBy: This edge denotes a relationship in which an entity(e,g artifact) is utilized by one or more entities(e.g process). An entity can use multiple entities so it is important to define the role.
- wasControlledBy: This edge outlines a relationship in which an entity caused the creation of another entity.
- used(Role): This edge denotes that an entity requires the services of another entity in order to execute.
- wasTriggeredBy: This edge denotes a relationship represents a process that was triggered by another process
- wasDerrivedFrom: This edge denotes a relationship which indicates that the source needs to have been generated before the destination is generated.

Objects represents the nodes of the relationship graph. There are three object contained in the OPM model: artifact, process, agent.

- artifact: An artifact represents the state of an entity. An artifact is graphically represented by a circle.
- Process: A process is an event which is taking place at a particular time due to an agent or an artifact. A process is represented by a square object.

- Agent: Agents represents actors that facilitate the execution of a process. An agent is represented by a hexagon in an OPM graph.

OPM is used to represent all previous and current actions that have been performed on an entity and the relationship between each entities contained in the graph. Figure 2 represents an example of an OPM acyclic graph with all of its causal dependencies.

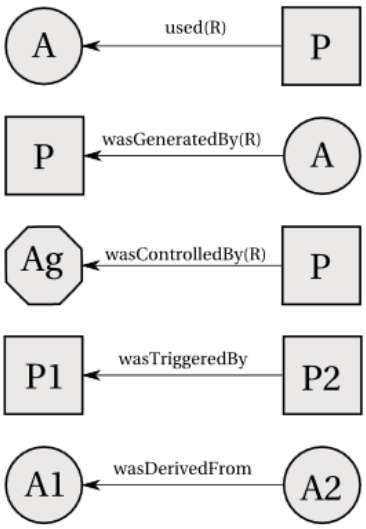

Figure 2.2: Edges and entities represented in OPM

2.4.2 Provenance Data Model(ProvDM)

Another model for representing provenance data is the Provenance Data Model(PROV-DM). PROV-DM is a W3C standardized extension of OPM. Prov-DM is a model that is used for depict causal relationship between entities, activities and , and agents(digital or physical). It creates a common model that allows for interchange of provenance information between heterogeneous devices. It contains two major components: types and relations.

- entity: An entity is a physical or digital object. An example of an entity is a file system, a process, or an motor vehicle.

- Activity: An activity represents some form of action that occurs over a time frame.Actions are acted upon by an entity. an example of an activity is a process opening a file directory, Accessing a remote server.

- Agent: An agent is a thing that takes ownership of an entity, or performs an activity. An example of an agent is a person, a software product, a process.

Figure below shows the various types contained in PROVDM and their representation. Entities, activities and agents are represented by oval, rectangle and hexagonal shapes respectively.

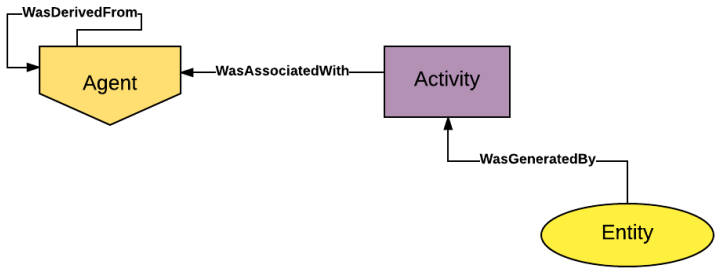

Figure 2.3: Prov-DM respresentation

Similar to the OPM, PROVDM does not keep track/estimate of future events. PROV relations are outlined below:

- wasGeneratedBy: This relation signifies the creation of an entity by an activity.

- used: This relation denotes that the functionality of an entity has been adopted by an activity.

- wasInformedBy: This relation denotes a causality that follows the exchange of two activities.

- wasDerievedFrom This relation represents a copy of information from an entity.

- wasAttributedTo: This denotes relational dependency to an agent. It is used to denote relationship between entity and agent when the activity that created the agent is unknown.

- wasAssociatedWith:This relation denotes a direct association to an agent for an activity that occurs.This indicates that an agent plays a role in the creation or modification of the activity.

- ActedOnBehalfOf: This denotes assigning authority to perform a particular responsibility to an agent. This could be by itself or to another agent.

ProvDM contains similar yet subtle differences between OPM.Some of the difference between OPM and PROV-DM are described below:

- The main components Artifact, Process and Agent in the OPM model are changed to Entity, Action, and Agent.

- Additional causal dependencies such as wasAttributedTo and actedOnBelafOf were included to represent direct and indirect causal dependencies respectively between agents and entities.

Since PROVDM is built on OPM and contains easy to understand constructs of types and relationships we are envision are properly represented through the various types and relations, we choose this model to represent provenance data in our IoT architecture instead of OPM.

2.4.3 PROV-JSON

PROVJSON is a component of PROVDM specification. It is used for representing PROV-DM data in JSON(Javascript Object Notation) format. It contains all of the components and relationships contained in PROVDM. It also allows for easy serialization and deserialization of PROVDM mappings to JSON representation. JSON is a lightweight, easy

to parse, and allows an easy way of manipulating data programmatically. An example of PROVJSON format is described below:

```
{
    "entity": {
        "e1": {
            "ex:cityName": {
                "$": "Londres",
                "lang": "fr"
            }
            ...
        }
    },
    "agent": {
        "e1": {
            "ex:employee": "1234",
            "ex:name": "Alice",
            "prov:type": {
                "$": "prov:Person",
                "type": "xsd:QName"
            }
        }
        ...
    },

    "activity": {
        "a1": {
            "prov:startTime": "2011-11-16T16:05:00",
            "prov:endTime": "2011-11-16T16:06:00",
            "ex:host": "server.example.org",
            "prov:type": {
                "$": "ex:edit",
                "type": "xsd:QName"
            }
        }
    }

}
```

Figure 2.4: PROV-JSON MODEL

2.5 Overview of Relevant Compression techniques

2.5.1 Lossy Compression

Lossy compression is a form of data compression in which the original data can be reconstructed with some tolerance of loosing some information contained in the original data.

This enables higher compression ratio than lossless compression techniques. Lossy Compression is mostly used in applications that does not have strict requirements as to loosing some information. An example of a lossless compression application can be seen in image compression software where the quality of the image is not seen by the naked eye. It is also used in speech transmission.

2.5.2 Lossless Compression

Lossless compression is a data compression technique in which the original data is reconstructed without loss of any information from. It can be used by applications which requires that the data compressed and the original data be identical. An example of such an application that requires lossless compression is text compression. Any compression that alters the structure of the original text results to a different meaning of the text. For instance, the sentence "I have a black cat" would have a different meaning if any information is lost from it.

Arithmetic Coding

Arithmetic coding is a form of lossless compression which encodes a stream of characters into a variable size interval between [0,1). A probability is assigned to each characters contained in the string. A cumulative probability is used to calculate the interval for the respective character. The more frequent characters are encoded with shorter codes than the less frequent characters.

Chapter 3

Related Work

This section outlines some of the state of the art in the area of data provenance collection systems, data compression techniques for data provenance, and models for representing provenance.

3.1 Related Work on Provenance Collection Systems

There has been a considerable amount of work done on data provenance collection. Some of this work has been focused on databases, sensor networks, scientific workflow systems and file system provenance but so far little attention has been given to provenance in IoT devices. Some the prior work done on data provenance collection are outlined below:

3.1.1 Provenance Aware Storage System(PASS)

MuniswamyReddy et al developed a provenance collection system that tracks systemlevel provenance of the 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, and provenance storage, 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 as inode catche. Provenance data is then transferred to a file system in an kernel database, BerkleyDB. This database maps key value pairs for provenance data for fast index look up. PASS collects and stores provenance information containing a reference to the executable that created the provenance

data, input files, A description of hardware in which provenance data is produced, OS information, process environment, process parameters, and other/data such as a random number generator seed.PASS detects and eliminates cycles that might occur in provenance dependencies as a result of version avoidance.Cycles violates the dependency relationships between entities. For example, a child node could depend on a parent node and also be an ancestor of a parent node. PASS eliminates cycles by merging processes that might have led to the cycles. It also provides functionalities for querying provenance data in the database. The query tool is built ontop of Berkley DB. For querying provenance, users can process queries using the provenance explorer. Query is done using commands such as MAKEFILE GENERATION which creates the sequence of events that led to the final state of a file or process. DUMP ALL, gets the provenance of the requested file.

3.1.2 HiFi

Bates et al. developed system level provenance collection framework for the Linux kernel using Linux Provenance Modules(LPM), this framework tracks system level provenance such as interprocess communication, networking, and kernel activities. This is achieved by mediating access to kernel objects. Linux Security Model is a framework that was designed for providing custom access control into the Linux kernel.It consists of a set of hooks which is executed before access decision is made. LSM was designed to avoid problem created by direct system call interception. The provenance information collected from the kernel space is securely transmitted to the provenance recorder in the user space.

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 log is a storage medium which transmits the provenance data to the user space.The collector contains the LSM which resides the kernel space. The collector records provenance data and writes it to the provenance log.The handler intercepts the provenance record from the log. This approach to collecting provenance data differs from our work since we focus on embedded systems and are concerned with input and

output (I/O) data, which primarily involve sensor and actuator readings.

3.1.3 RecProv

RecProv 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. Mozilla rr is a debugging tool for linux browser. It is developed for the deterministic recording and replaying of the firefox browser in linux. Recprov uses PTRACE_PEEKDATA from PTRACE to access the derefrenced address of the traced process from the registers. It also ensure the integrity of provenance data up till the point at which a host is compromised by trace isolation. Mozilla rr relies on PTRACE which intercepts system calls during context switching.It uses PTRACE to monitor the CPU state during and after a system call. It uses PTRACE_PEEKDATA from PTRACE to access the derefrenced address of the traced process from the registers.System calls such as execve, clone, fork, open, read, write, clode, dup, mmap, socket, connect, accept are monitored which is used for file versioning. The provenance information generated is converted into PROV-JSON, a W3C standard for relaying provenance and also stores provenance data in Neo4j a graph database for visualization and storage of provenance graphs.

3.1.4 StoryBook

Spillance et al developed a user space provenance collection system, Storybook which allows for the collection of provenance data from the user space thereby reducing performance overhead from kernel space provenance collection. This system is modular; It allows the use of application specific extensions allowing additions such as database provenance, system provenance, and web and email servers. It achieves provenance capture by using FUSE for system level provenance and MySQL for database level provenance capture. StoryBook

allows developers to implement provenance inspectors these are custom provenance models which captures the provenance of 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, causality relationship between entities contained in the provenance system. Provenance data is stored in key value pairs using Fable as the storage backend. Storybook allows for provenance query.It achieves this by looking up an inode in the file, ino hashtable.

3.1.5 Trustworthy whole system provenance for Linux kernel

Bates et al provide a security model for provenance collection in the linux kernel. This is achieved by creating a Linux Provenance Model(LPM). LPM serves as a security layer that provides a security abstraction for provenance data. It is involved in the attestation disclosure of the application layer, authentication channel for the network layer. The goal of LPM is to provide an end to end provenance capture system. LPM ensures the following security guarantees: For LPM,the system must be able to detect and avoid malicious forgery of provenance data. The system must be trusted and easily verifiable.

When an application invokes a system call, the LPM tracks system level provenance which is transmitted to the provenance module via a buffer . The provenance module registers contains hooks which records system call events. These events are sent to a provenance recorder in the user space. The provenance recorder ensures that the provenance data is stored in the appropriate backend of choice. Provence recorders offer storage for Gzip, PostGreSQL, Neo4j and SNAP.

3.1.6 Towards Automated Collection of Application-Level Data Provenance

Tariq et all developed a provenance collection system which automatically collects interprocess provenance of applications source code at run time. It takes a different approach from system level provenance capture. It achieves provenance capture by using LLVM compiler framework and SPADE provenance management system. LLVM is a framework that allows dynamic compilation techniques of applications written in C, C++, Objective-C, and Java. Provenance is inserted during compilation. LLVM contains an LLVM Reporter. This is a java class that parses the output file collected from the LLVM tracer and forwards the provenance data collected to the SPADE Tracer. SPADE is a provenance management tool that allows for the transformation of domain specific activity into provenance data. The LLVM Tracer module tracks provenance at the exit and entry of each function. The output is an Open Provenance Model in which functions are represented by an OPM process, arguments and return values are represented as an OPM artifact. To minimize provenance record, users are allowed to specify what functions they would like to collect provenance information at run time. The system discards provenance data that are not needed. The LLVM optimizer generates a graph of the target application. This graph is used to perform reverse reachability analysis from the functions contained in the graph. A workflow of how the framework works in collecting provenance is as follows:

- The application is compiled and converted into bitcode using the LLVM C compiler, clang.
- The LLVM Tracer module is compiled with bitcode.
- Provenance instrumentation is added to the bitcode via the LLVM Trace module.
- instrumentation bitcode is converted into assembly code via the compiler llc
- LLVM Tacer and Reporter is compiled into assembly code.

- The instrumented application and assembly code is compiled into an executable binary and sent to the SPADE kernel.

One major limitation to this approach of collecting application level provenance is that a user is required to have the source code of the specific application in which provenance data is required. Also, provenance collection is limited to function's exit and entry points.

3.1.7 Provenance from Log Files: a BigData Problem

Goshal et al developed a framework for collecting provenance data from log files.They argue that log data contains vital information about a system and can be an important source of provenance information. Provenance is categorized based on two types: process provenance which involves provenance information of the process in which provenance is captured. Data provenance describes the history of data involved in the execution of a process. Provenance is collected by using a rule based approach which consists of a set of rules defined in XML. This framework consists of a rule engine. The rule engine processes raw log files to structured Provence. There are three categories of rules as specified by the grammar. The match-set rules which selects provenance data based on a set of matching rules. Link Rules which specifies relationship between provenance events and Remap rules are used to specifying an alias for a provenance object.T he framework consists of three major components. The Rule engine which contains XML specifications for selecting, linking and remapping provenance objects from log files. This component is integrated with the log processor. The Log processor component is involved in parsing log files with information contained in the rule engine. It converts every matching log information to a provenance graph representing entities(vertices) and their relationship(edges). The adaptor converts the structured provenance generated by the Log processor into serialized XML format which is compatible with karma, a provenance service application that is emplyed for the storage and querying of the provenance data collected.

3.1.8 Towards a Universal Data Provenance Framework using Dynamic Instrumentation

Gessiou et al propose a dyamic instrumentation framework for data provenance collection that is universal and does not incur the overhead of kernel or source code modifications. This is achieved by using DTrace, a dynamic instumentation utility that allows for the instrumentation of user-level and kernel-level code and with no overhead cost when disabled. It allows to be included at every point in time in a system runtime without overhead issues from disabled probes.

The goal is to provide an easy extension to add provenance collection to any application regardless of its size or complexity. They implement the provenance collection scheme on file system using PASS, on a database(SQLite) and a web browser(Safari). The logging component monitors all system calls for all processes involved. It contains information such as system-call arguments, return value, user id, process name, process if and timestamp. The logging components include functionalities to collect library and functional calls.They argue that applying data provenance to different layers of the software stack and not just the system level can provide varying levels of information. Provenance needs to be captured at the layer in which it is closest to the information that is needed to be captured. They collect provenance information that pertains to complex systems activities such as a web browser or a database. The information is collected from a process system-call and functional-call based on valid entry points contained in the process. It allows information from multiple entry points. DTrace dynamic instrumentation helps in discovering interesting paths in a system in which data propagates. This helps users use this information to collect a more accurate provenance data.

3.1.9 Provenance for Sensors

Lim et al. developed a model for calculating the trust of nodes contained 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 affected by the tust score of the sensor that forwards data to the system. Provenance is determined by the path in which data travels through the sensor network. This work differs from our approach since the authors focus on creating a trust score of nodes connected in a sensor network using data provenance and do not emphasize how the provenance data is collected. We are focused on creating a secure provenance aware system for I/O operations which is used to ensure trust of connected devices.

3.1.10 Provenance Recording for Services

Grouth et al developed a provenance collection system, PReServ which allows software developers to integrate provenance recording into their applications mainly focused on scientific applications. They introduce P-assertion recording protocol as a way of monitoring process documentation for Service Oriented Architecture(SOA) in the standard for grid systems. PreServ is the implementation of PreP. PreP specifies how provenance can be recorded. PreServ contains a provenance store web service for recording and querying of p-assertions. P-assertions are provenance data generated by actors about the execution. It contains information about the messages sent and received as well as state of the message. PreServ is composed of three layers, the Message Translator which is involved with converting all messages receive via HTTP request to XML format for the provenance store layer. It also determine which plug-in handle to route incoming request, the Plug-Ins layer receives messages from the Message translator. It implements functionality that the Provenance Store component provides. This component allows third party developers to add new functionality to store provenance data without going through details of the code implementation. The backend component which stores the P-assertions. Various backend implementations could be attached to the system by the plug-in component

3.2 Related Works on Data Compression

3.2.1 Secure Data Provenance Compression Using Arithmetic Coding in Wireless Sensor Networks(WSN)

The authors encode the provenance of the path in which a sensor provenance travels using arithmetic coding, a data compression algorithm.Arithmetic coding assigns intervals to a string of characters based on cumulative probabilities of each character contained in the string. It assign probabilities by monitoring the activities of the WSN, this is known as the training phase. In this phase the number of times a node appears in a packet is counted. Each node in the WSN is regarded as a symbol. This is used as input to the arithmetic coding algorithm which generates a variable interval for the combination of symbols. For their application, provenance is referred to as the path in which data travels to the BS. The WSN contains a Base Station(BS) and a network of nodes. The base station is in charge of verifying the integrity of the packets sent. It is also where the encoded characters using arithmetic coding is decoded. According to the authors, provenance into two types: Simple provenance in which data are generated from the leaf nodes and are forwarded to surrounding nodes towards the Base Station. In the other form of provenance, data is aggregated at an aggregation node and forwarded to a BS. Each node in the WSN is represented as a string of characters and is encoded using arithmetic coding. The BS receives the encoded provenance and decodes it. Provenance is encoded at the respective nodes and forwarded to the next node in the graph and it is decoded in the BS. The BS recovers the characters the same way they were encoded. It also receives the encoded interval to which it uses to decode characters by using the probability values saved in its storage.

Chapter 4

Sensor Data Provenance Collection Framework for the Internet of Things

In this chapter, we define how provenance is collected and modeled along the IoT architectural stack. We also define implementation specifics of the provenance collection framework for IoT device.

4.1 IoT ProvenanceCollection framework Information flow

A provenance model is used to represent causal relationship between objects and it is usually modeled as a Directed Acyclic Graph(DAG).This enables for better graphical representation of provenance relationships and also a unified format for representing provenance data. There are two major models for representing provenance, OPM and ProvDM. PROVDM is a predecessor and standardized version of OPM. It allows the modeling of data objects either physical or digital. PROVDM is chosen as the model to represent provenance for our implementation because it allows for proper representation of all of the relationships in which we envision for IoT devices. This section defines a model for relaying provenance in IoT systems which is built on top of PROVDM. From the IoT architecture as described in Section 2, we can see that data is disseminated from sensors and actuators across various layers contained in the IoT architectural stack. The provenance data produced from various sensors is collectively aggregated at the gateway layer and/or the cloud layer. We allow

for provenance data to be translated to the appropriate PROVDM format at the various levels of the IoT architecture. This allows for fault tolerant processing of information even in the case of a network failure at all layers. Sensor readings are collected from devices as specified by a policy. This helps address the issue with memory constraint of collecting provenance data in IoT devices. Policy specification and implementation are discussed in greater detail in chapter 4. Figure 4.1 illustrates the data aggregation at various layers of the IoT architectural stack.

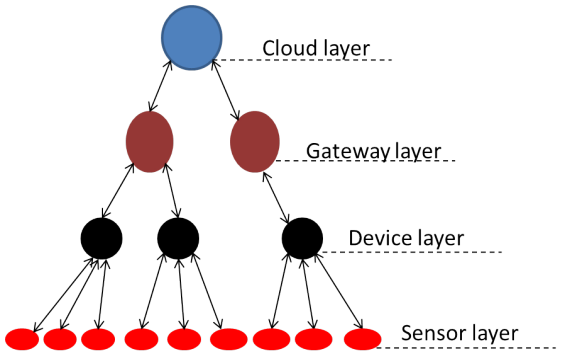

Figure 4.1: IoT provenance-collection data aggregation

Using the example of a smart home use case as described in chapter 2, a detailed example of how our provenance collection framework can be applied is described below

- Provenance data is collected from sensors and actuator readings of devices contained in the smart home(e.g thermostat, refrigerator, and smart doors).The provenance data is collected as specified in a policy document. Policy document contains information of which provenance data to be collected in the device and can be only be defined by the device owner(e.g house owner).

- Provenance data from each device is aggregated and passed along to the gateway. The information is further analyzed and transmitted to the cloud for storage in which further analysis could be conducted on the data to derive more insights from the provenance data collected. The gateway collectively aggregates data from multiple

sensors and forwards the aggrregated data to the cloud for further processing. This information is then transferred to the cloud where it is mapped using the PROVDM which allows for causal relationship between the objects contained in the IoT framework. The cloud contains all of the aggregated provenance data from the respective devices contained in the smart home in which further analysis can be performed on the collective provenance data.

Each layer in the provenance IoT architecture is indepent of the and maintains provenance information that can be mapped using the PROVDM format which allows for the representation of dependencies that exists between objects contained in the device. This allows for provenance data to be further analyzed at the respective layers even in an event of network loss.

4.1.1 Provenance-Collection Model Definition

It is assumed that provenance data is collected from the underlying IoT device using the framework outline in section 3 of this chapter below. Data is collected from sensors and actuators attached to an IoT device.The underlying construct is represented as a acyclic graph which denotes the relationship between multiple sensors and actuator readings. Since PROV-DM type provides a generic model for relaying provenance information, we define a more specific construct for representing provenance data in IoT devices based on PROV-DM. In the context of IoT provenance entity, process and agent are defined below:

- Agent: An agent contains information on an object that accesses an entity. Examples of agents in an IoT architecture are sensors, actuators, user roles(e.g admin). A unique identifier is given to each agents contained in the IoT framework.
- Entity: An entity can be defined as a data object that contains information which can be modified. An example entities are device files, processes, device memory contents.
- Activity: An activity is a modification that an agent makes on an entity. An example

of an activity are basic file access modifications such as read, write, delete, update.

To better emphasize the provenance-collection model, an example of a use case for the provenance collection model is illustrated in the figure below.

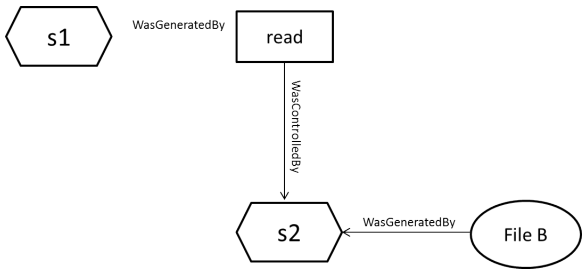

Figure 4.2: Provenance-model use case

The figure depicts a dependency relationship between two sensors, s1 and s2. Consider the scenario of a smart home in which, s1 is smart thermostat contained in a smart home which regulates the temperature of the home and s2 is a sensor that detects the temperature outside of the house. s1 constantly checks the temperature outside to regulate the temperature of the house accordingly. s1 tries to access information from s2. According to the provenance data model definition, s1 and s2 are agents. The activity performed on s2 by s1 is read. File B is the entity in which s1 tries to read from s2 to determine the environment temperature. The relationship between various components contained in the model is illustrated on the edges of the graph.

The relations which defines the relationship between types of a provenance model. We use the same instance of the relations contained in prov-dm to represent relationships between types contained in the IoT framework.

4.2 Provenance-Collection System

In this section, we outline the components of our system and describe how provenance trace is collected across the IoT framework. Figure 1 displays the system architecture of

our approach. Sensor and actuator readings in the form of I/O trace are recorded by the tracer component. This component intercepts system level I/O events and produces trace information in the Common Trace Format (CTF). CTF represents binary trace information containing multiple streams of binary events such as I/O activity. Trace information is converted into the Open Provenance Model (OPM), which represents the relationship between provenance entities contained in the system. Our system relies heavily on data pruning to reduce and remove unimportant provenance in order to conserve memory. This is achieved by a policy based approach in which a user with administrative privileged is given permission to create or modify a policy. This policy document dictates what provenance data to collect. Pruned provenance from the device is securely transmitted to a gateway and later transmitted and stored in a private cloud backend.

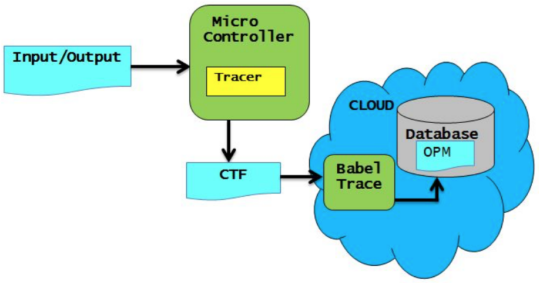

Figure 4.3: Proposed Model

The goal is to create a provenanceaware system that records I/O operations on data for devices connected in an IoT system. For our implementation, several tools and hardware components are utilized in the development of our prototype, some of the tools utilized are outlined below:

- BeagleBone Black Board. This device is a microcontroller used to evaluate our approach. We choose the BeagleBone Board because it is a representation of what can be found on an IoT gateway device and it has the capability to include custom

hardware in programmable logic. Also, BeagleBone is a lowcost, simple IoT demonstrator that was chosen for its highperformance, onboard emulation, and IoT gateway projects can be programmed without additional need for hardware tools.

Chapter 5

Storage Optimization Framework for Provenance data storage in IoT devices

In this chapter we define our model and algorithms for storage optimization of provenance data in IoT devices. We focus on all of the proposed techniques such as data pruning and data compression of graphs. We evaluate our hypothesis the proof of concept ideology.

We envision large amounts of provenance data will be generated from our provenance collection framework. Due to the memory constraints on devices at the sensor and device layer of the IoT architecture, there is a need to create for an efficient storage of provenance data. To address this issue, we define a policy driven framework that provides a policy scheme for the enforcement of provenance storage. The policy framework contains a policy document which contains information in the provenance that are permitted to be collected. A policy framework is used to address the resource constraint problems encountered when storing large amounts of provenance data on low memory devices. Policy document is created by the user who serves as an administrator. An administrator is a user has the right to add, delete or modify the policy document. For implementation, there are several policy models can be utilized to integrate a policy model with the IoT framework. To implement a policy based framework, We propose to use the eXtensible Markup Language(XACML) policy model to to generate and enforce provenance policy for our IoT framework. XACML is an access control policy based language that allows for the creation and enforcement of policies written in XML. Since it is based on XML, this framework allows for extensible and

flexibility of policy documents. Compression techniques could be applied at various levels of the IoT architecture to provide further storage efficiency of provenance data. Using the use case of the smart home depicted in chapter 2, a policy framework could be implemented and incorporated into the IoT framework which allows a device administrators to specify what kinds of provenance data to collect. The policy acts an enforcement point providing an efficient storage mechanism in a resource constrained environment of the devices contained in the smart home.The contents of the implementation details for the policy framework which specifies the policy grammar and the policy architecture across the IoT stack are left as future work.

Chapter 6

Conclusion

6.1 Summary

In this research, we motivate the need for provenance into the IoT architecture. We propose a provenance collection system that provides provenance collection capabilities for devices contained in an IoT framework . Provenance data is believed to be beneficial in mitigating current and future malicious attacks. To address the issue of provenance storage, we also propose a framework for efficient provenance data storage.

6.2 Future Work

Some of the proposed future work for this research is outlined bellow:

- Provenance collection raises privacy issues.How do we ensure that the vast amount of data collected is not invasive to the privacy of the use and also is not used as a tool to perform malicious attacks.
- Provenance Query: How do we query provenance data in order to interprete provenance information and make anaylsis of data collected.
- Provenance Versioning: Provenance version creates cycles. When a file is read or edited do we create a new instance of the file? Tracking all transformations that occurs on a data object in memeory constrained devices might lead to running out of storage space. A new instance of the provenance of that file should be created and included in the provenance data.

- Securing Provenance: Due to the great level of sensitivity that provenance data entails, it is of utmost importance to ensure the confidentiality and integrity of provenance data stored on IoT devices while at rest or in transit. Proper encryption and authentication techniques need to be employed to ensure the confidentiality, integirty, and availability of provenance data.