Fear and Logging in the Internet of Things

Qi Wang, Wajih Ul Hasan, Adam Bates, Carl Gunter
University of Illinois at Urbana-Champaign

Published at NDSS 2018

Presented By
Md Mahbubur Rahman
Computer Science, Wayne State University

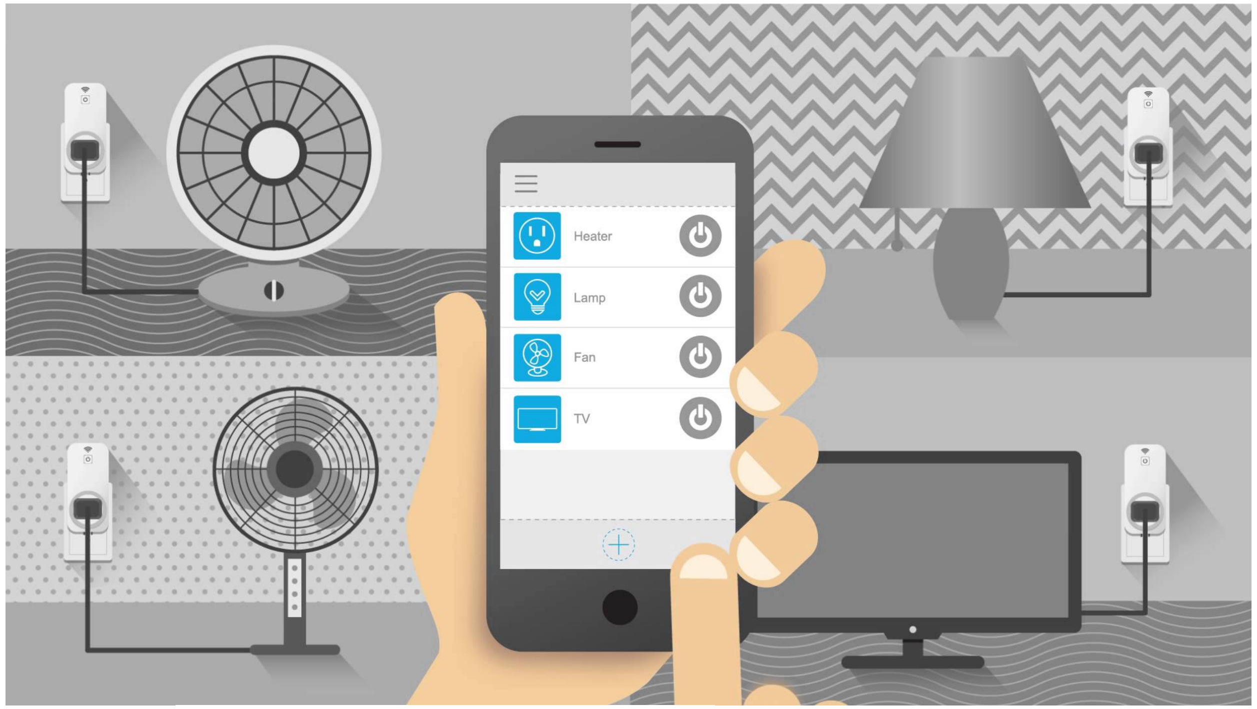
September 24, 2018

Outline

Internet of Things (IoT)

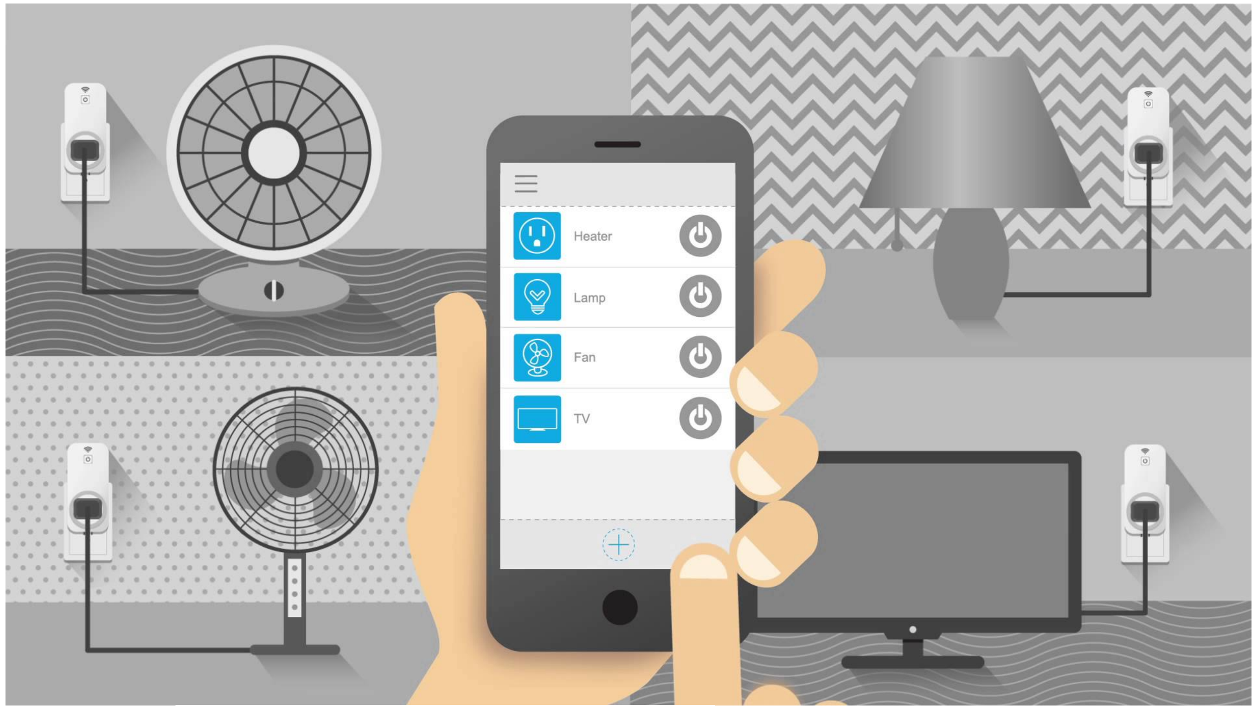
- A network of interconnected devices/sensors
  - Devices can exchange data via a common interface
  - Interface is connected to the Internet
- As of 2017, the number of IoT devices increased to 8.4 billion
  - By 2020: 30 billion devices
  - By 2020: Market value of IoT is projected to reach $7.1 trillion
- Example: Smart Home
  - Lock/unlock your door with a smart phone application

A Smart Home



Source: TheAutomization.com

A Smart Home



Source: The Automization.com

SmartThings

wink

amazon alexa

HomeKit

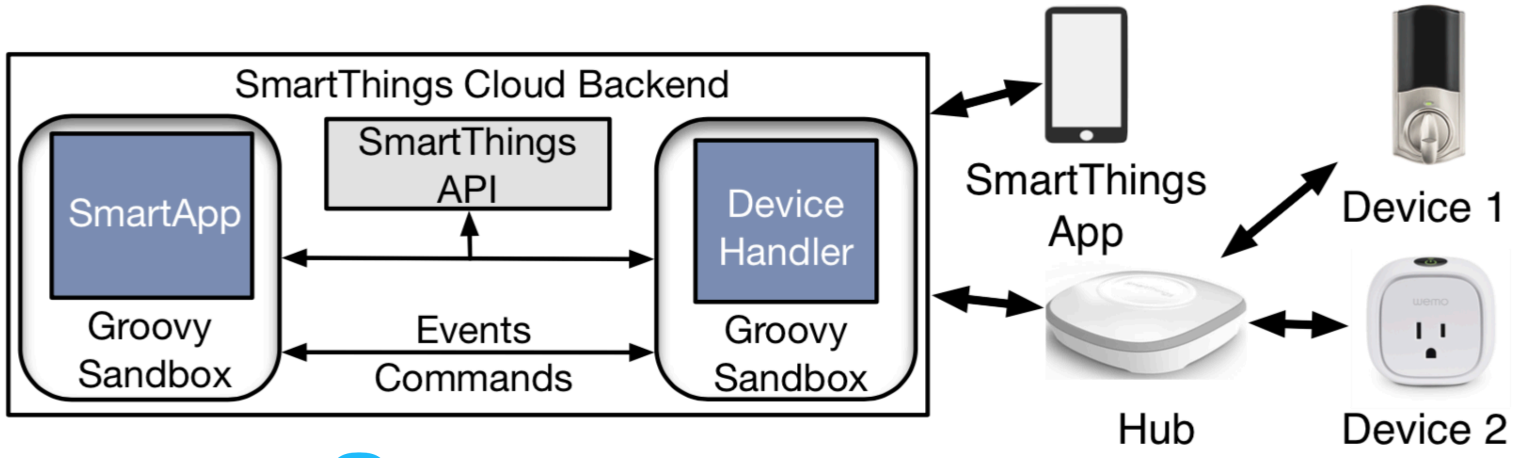
IRIS
COLLECT • COMMUNICATE • CONTROL
SECURE IoT SOLUTIONS

450+ other vendors!!!

Common Architectures

- All the devices are connected to a **Hub**
- A **Cloud** synchronizes device states and provide interfaces for remote monitoring
- An **App** is a program that manages devices

Hub-centric & Cloud-centric Architectures



SmartThings

Cloud-centric, but have a **Hub** as well.

Security Concerns

- How to diagnose an **incorrect/malicious/misconfiguration** behaviors
  - Trigger-action programming can create a chain (flow) of devices and apps together to the point that determining the root cause of an unexpected behavior/event is often difficult.
  - Malicious IoT apps may exists in a chain.
  - A malicious app may forge a CO detection event and an alarm detection app may sound the alarm because it cannot detect the illegitimate history of the event.
- How to explain the overall system behaviors?
- Need to understand the lineage of triggers and actions that occurs.

Logging in IoT Platforms

- Current logging mechanism in IoT is **device-centric**
  - It is difficult to create a causal dependencies between different events and data states

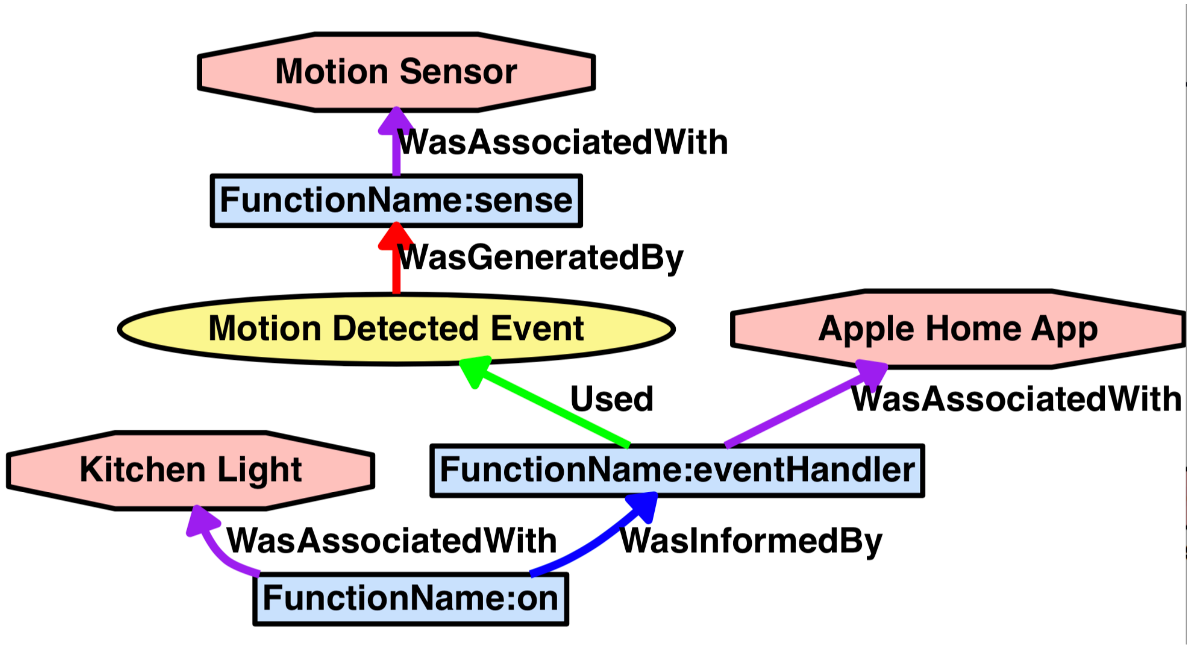
- Authors analyzed the logs of an Iris System
  - *“Motion was detected by Iris indoor camera at 11:13 AM”*
  - *“Front door was unlocked at 11:13 AM”*
  - *“Light was turned on at 11:14 AM”*

Why the light was turned on at 11:14 AM?

Data Provenance

- Describes the **history of actions** taken on a data object from its creation up to the present
  - “In what environment was this data generated?”
  - “Was this message derived from sensitive data?”

The light was turned because motion was detected



Provenance of Apple HomeKit

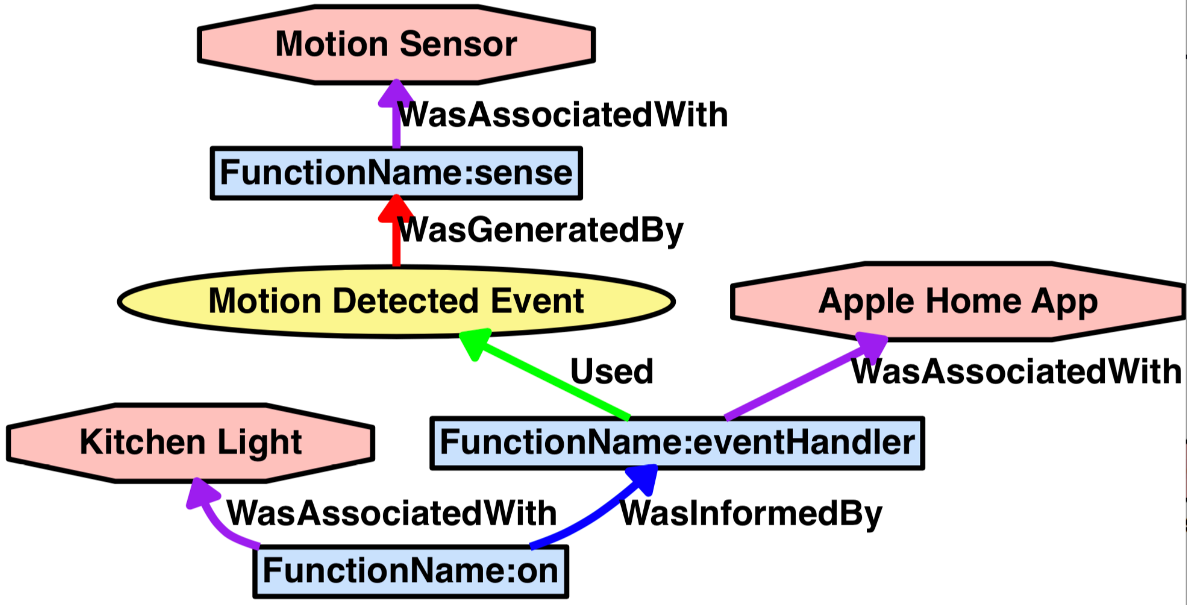
Tool: W3C PROV-DM
Its pervasive and represents provenance graph in a DAG

PROV-DM [1]

- PROV-DM has three types of nodes
  - **Entity**: is a data object
  - **Activity**: is a process
  - **Agent**: is something that is responsible for Entities and Activities

- **Edges**: encode dependency types between nodes

Which Entity **WasAttributedTo** which Agent
Which Activity **WasAssociatedWith** which Agent
Which Entity **WasGeneratedBy** which Activity
....



Provenance of Apple HomeKit

1. https://www.w3.org/TR/prov-overview/

ProvThings: A Framework

- Threat Model & Assumptions
  - **API-level attacks**: attacker is able to access or manipulate the state of the smart home through creation and transition of well-formed API control messages.
  - **Accidental App configuration**

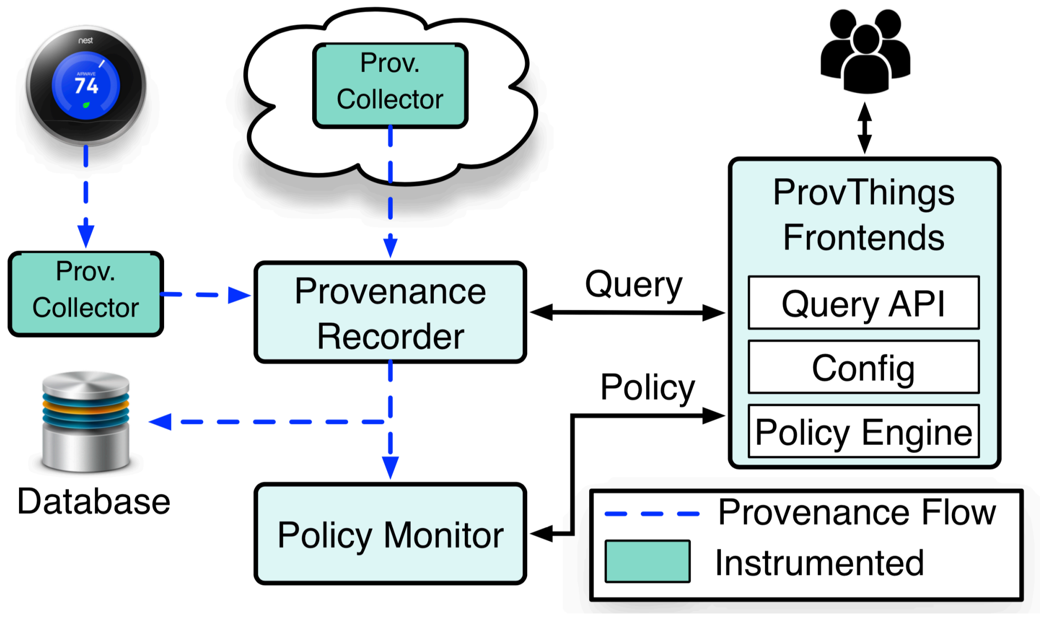
- Plausible scenarios through which API-level attacks may happen
  - Malicious Apps
  - Device Vulnerabilities
  - Proximity

ProvThings: A Framework

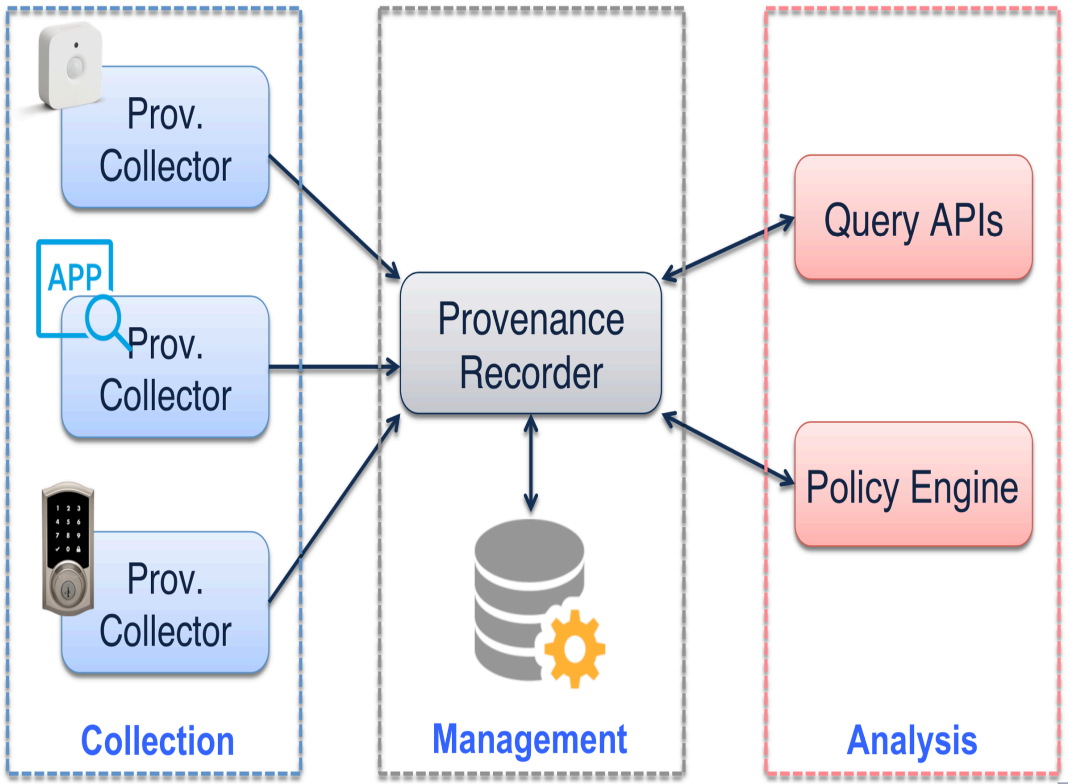
- Assumptions
  - Attacker cannot get the root access of the devices
  - Attacks through communication protocols are out of scope
  - Entity responsible for IoT central management is not compromised
    - SmartThings Cloud

ProvThings: Overview

- ProvThings is a general framework for **collection**, **management**, and **analysis** of data provenance in IoT platform



Architecture of ProvThings provenance management system

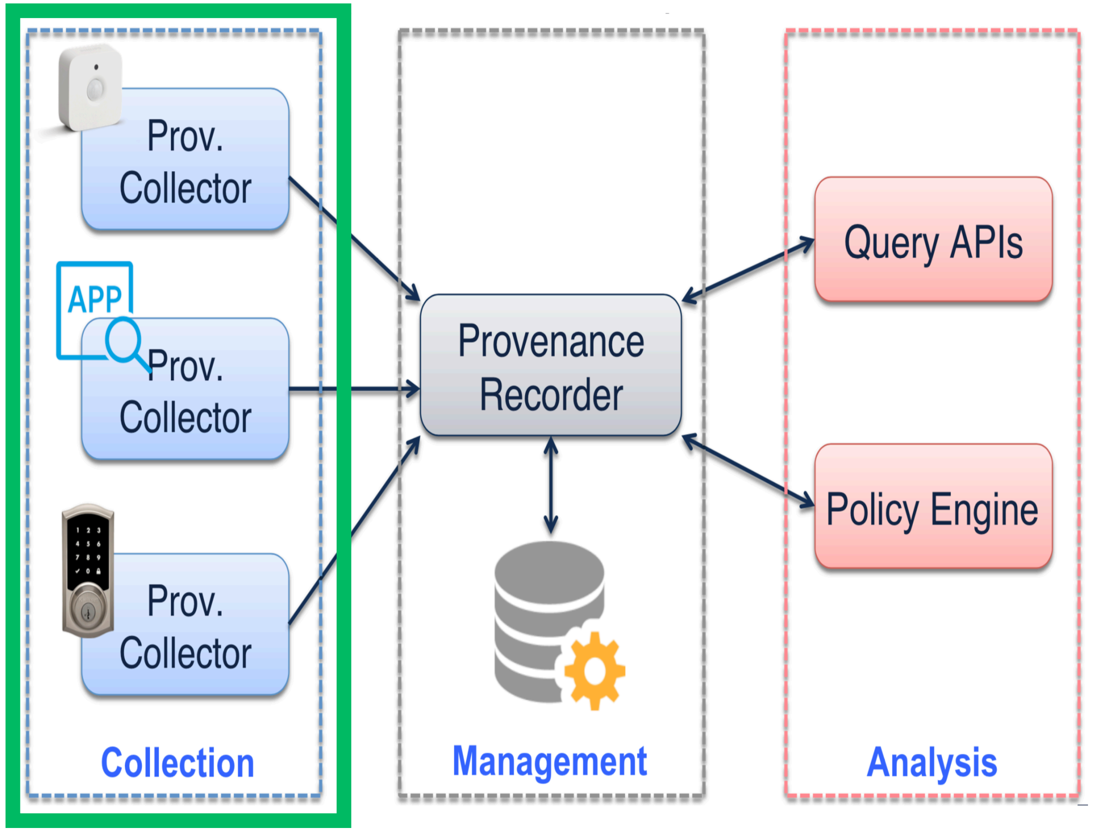


Courtesy: the Authors

Provenance Collection

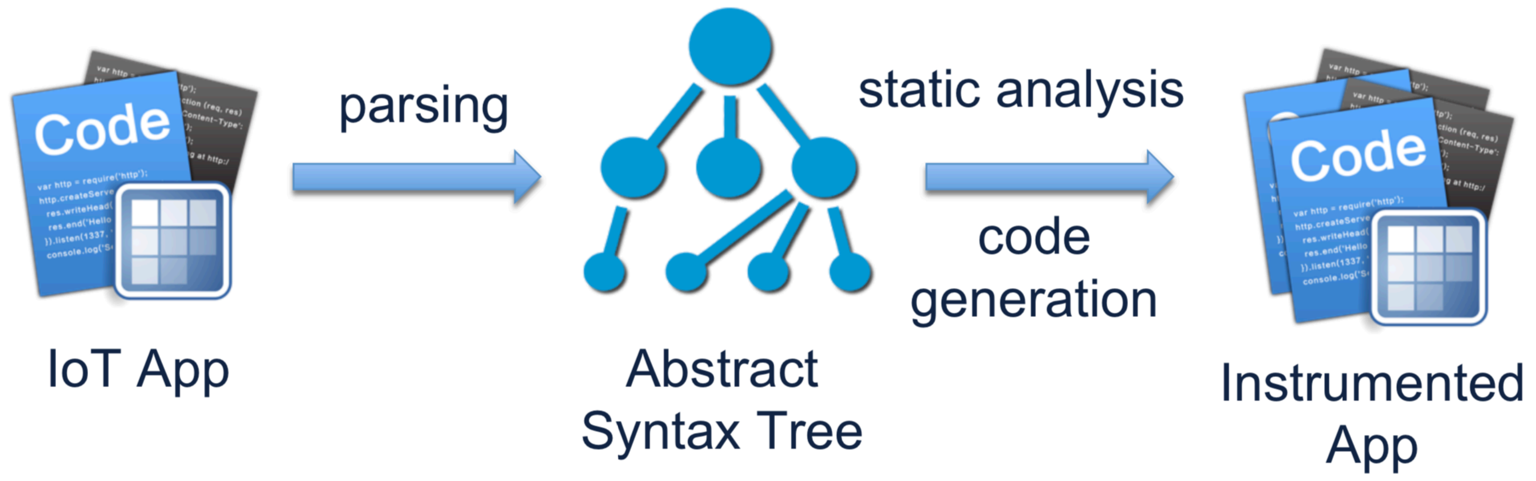
- ProvThings collect provenance metadata from different components of an IoT platform
  - **IoT Apps**
  - **Device Handlers**

- Uses **automated program instrumentation** to collect metadata
  - Minimally invasive since it does not do any hardware instrumentation



Program Instrumentation

- ProvThings instruments IoT Apps statically
  - Helps build the **control flow** and **data flow**

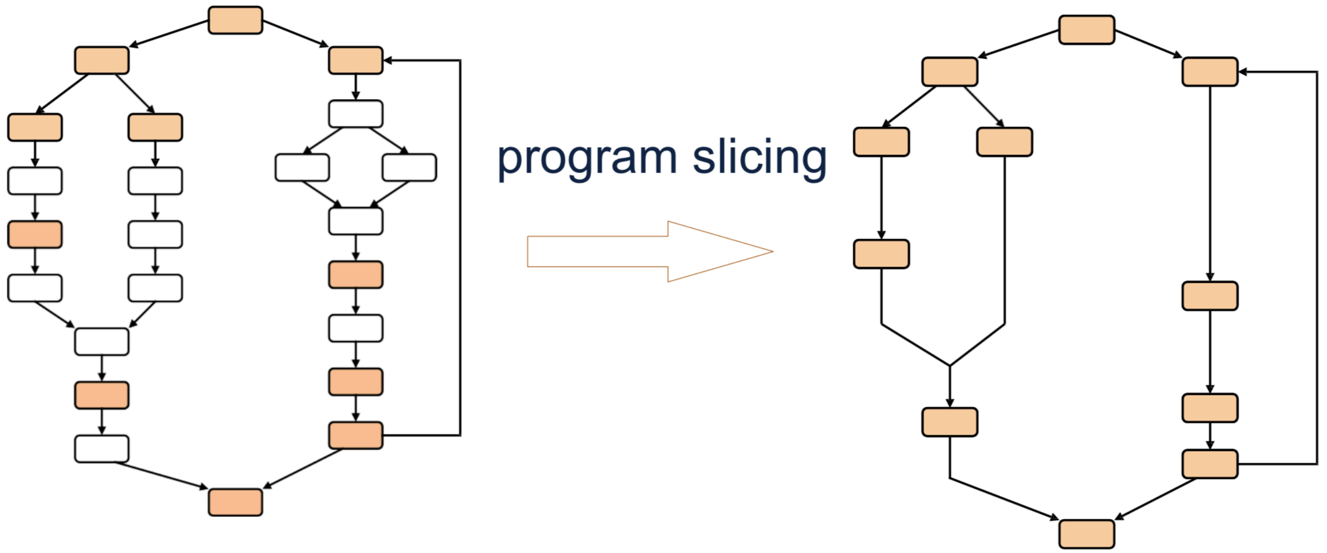


Courtesy: the Authors

- Instrumented App/code collects provenance metadata at runtime

Selective Program Instrumentation

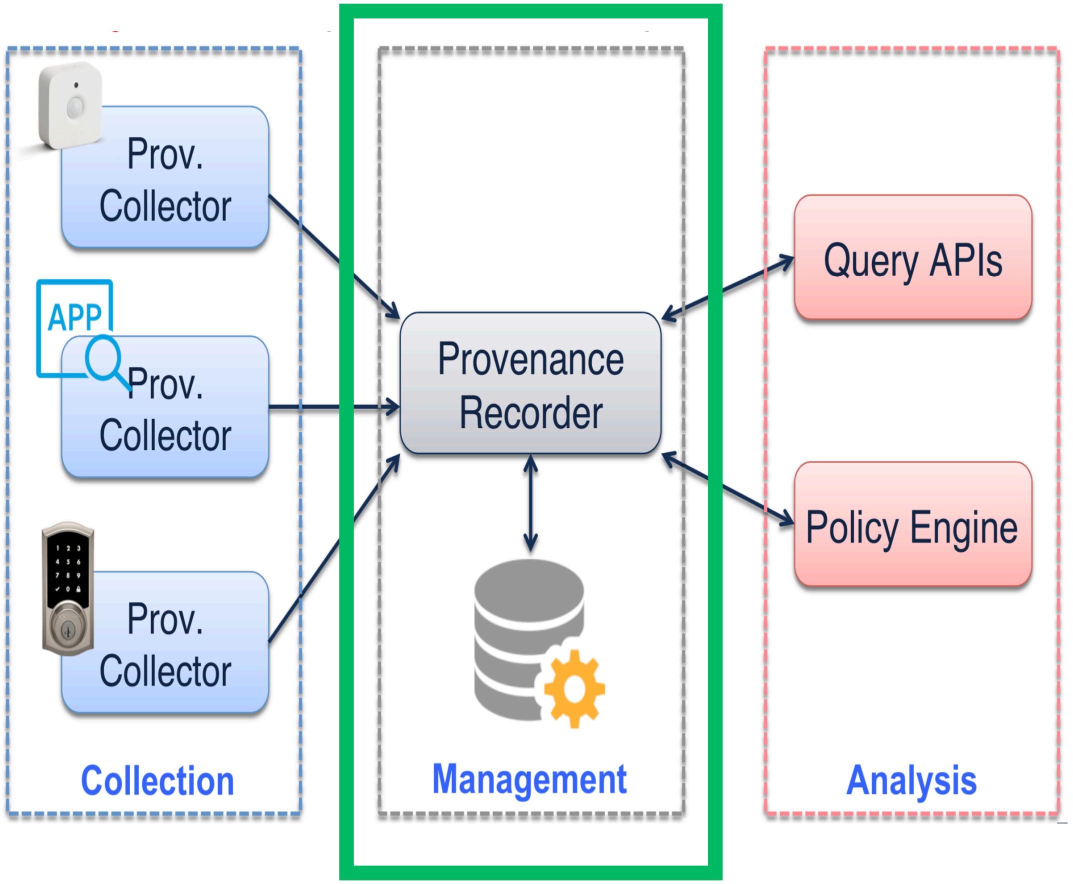
- Helps to avoid collecting unnecessary provenance metadata
- Define provenance in terms of **Sources** and **Sinks**
    - **Source**: a security sensitive data object (e.g., state of a lock)
    - **Sink:** a security sensitive method (e.g., command to unlock a door)



Courtesy: the Authors

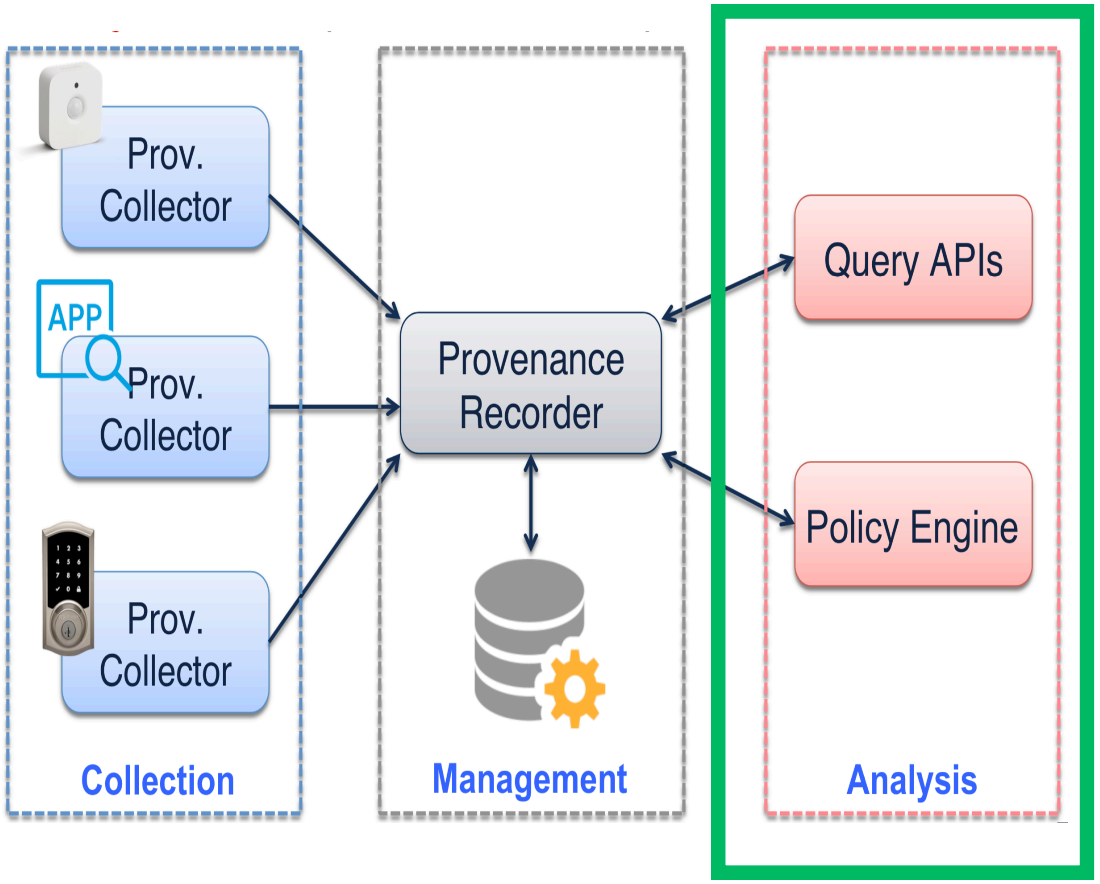
Provenance Management

- Aggregates and merges provenance records from different collectors, filters them, and converts them into a unified IoT provenance model
- Builds and stores the provenance graph in a database
  - Adds modular support for different backends: SQL, Neo4j.



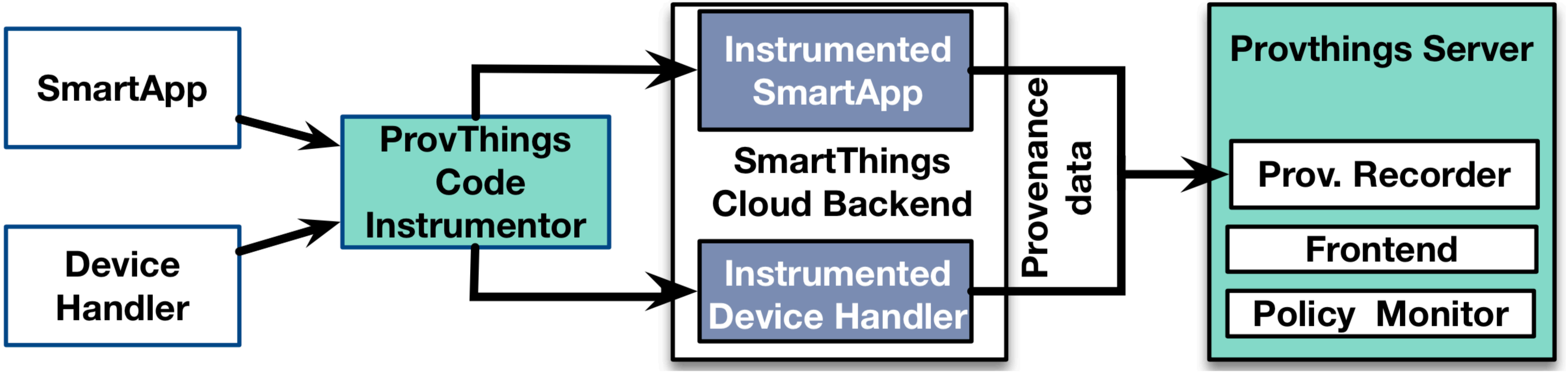
Provenance Analysis

- **Query APIs**: can analyze forward and backward dependency analysis
- **Policy Engine**: allows users to create configuration, policies in the form of graph
- **Policy Monitor**: Cross-checks with provenance graph if it's a valid policy or not

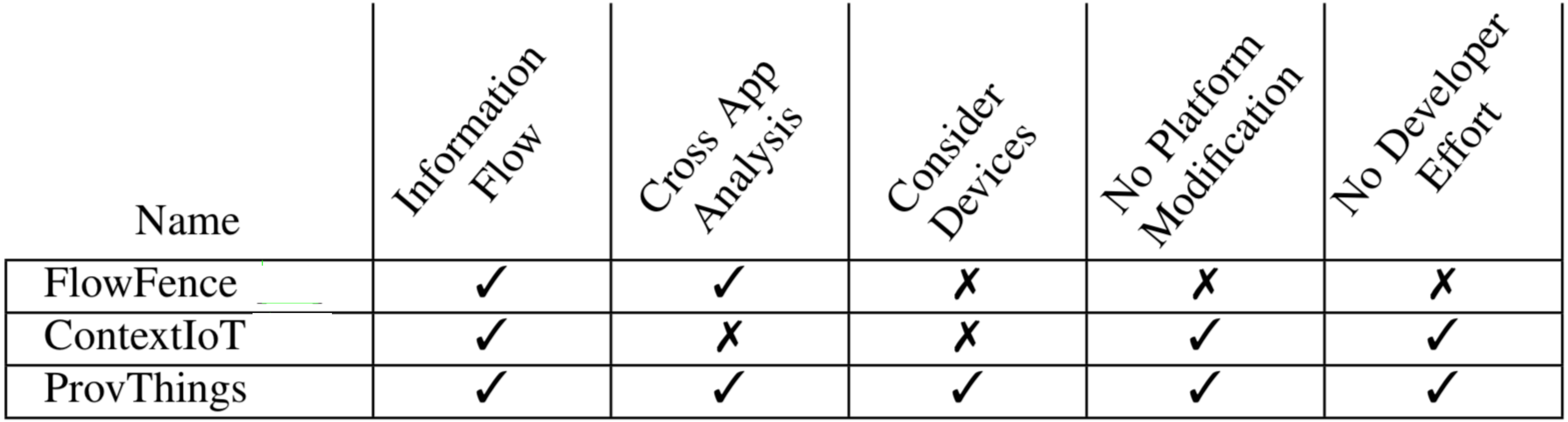


Implementation

- Implemented on top of Samsung SmartThings



Implementation: Comparison



Name	Information Flow	Cross App Analysis	Consider Devices	No Platform Modification	No Developer Effort
FlowFence	✓	✓	✗	✗	✗
ContextIoT	✓	✗	✗	✓	✓
ProvThings	✓	✓	✓	✓	✓

Evaluation

- Evaluate on five metrics
    1. Effectiveness of attack reconstruction
    2. Instrumentation overhead
    3. Runtime overhead
    4. Storage overhead
    5. Query performance

- Evaluation of 1 and 3 is done at SmartThings IDE cloud
- 2, 4, and 5 is evaluated at a local machine with Intel Core i7-2600 Quad-Core 3.4GHz processor with 16GB RAM running Ubuntu

Evaluation

- Overhead measurements
  - Unmodified (vanilla) SmartApps
  - ProvFull (instruments all instructions to collect provenance data)
  - ProvSave (Apply selective code instrumentation)

- Dataset
  - SmartApps of 26 possible IoT attacks [2]
  - 236 commodity SmartApps

2. ContexIoT, *Jia et al. NDSS' 17*

Evaluation

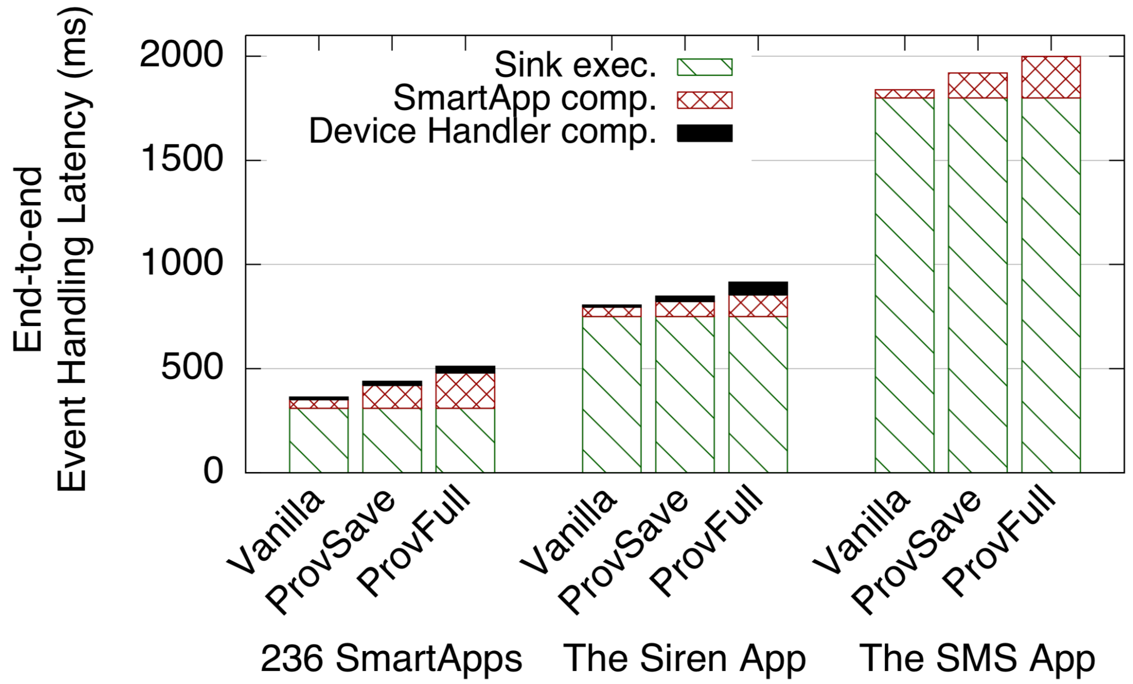
- ProvThings were able to effectively reconstruct all 26 attacks
- **34ms** for SmartApps and **27ms** for device handlers as the instrumentation overhead
- **260KB** of daily storage overhead

2. ContexIoT, *Jia et al. NDSS' 17*

Evaluation

- **End-to-end latency** on event handling due to provenance collection
  - An event handler sends a text message if motion is detected by a motion sensor, the end-to-end event handling latency is the time between the motion event is received and the time message is delivered to the user.

Tested on both virtual and physical devices

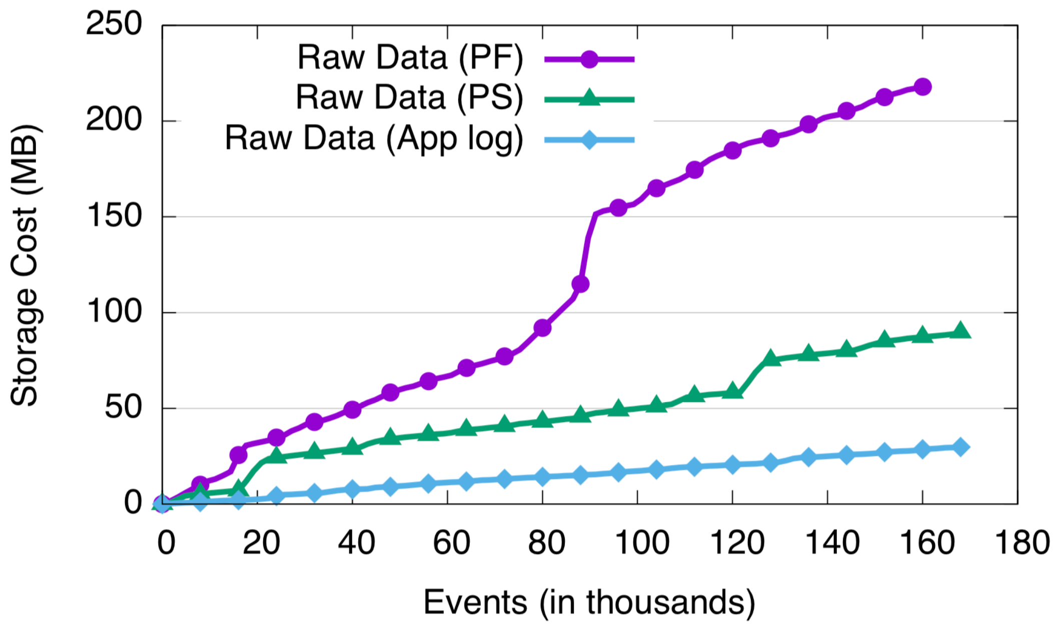


In simulation
ProvSave: 20.6% overhead
ProvFull: 40.4% overhead

Real Devices
ProvSave: 5.3% and 4.5% overhead
ProvFull:13.8% and 8.7% overhead

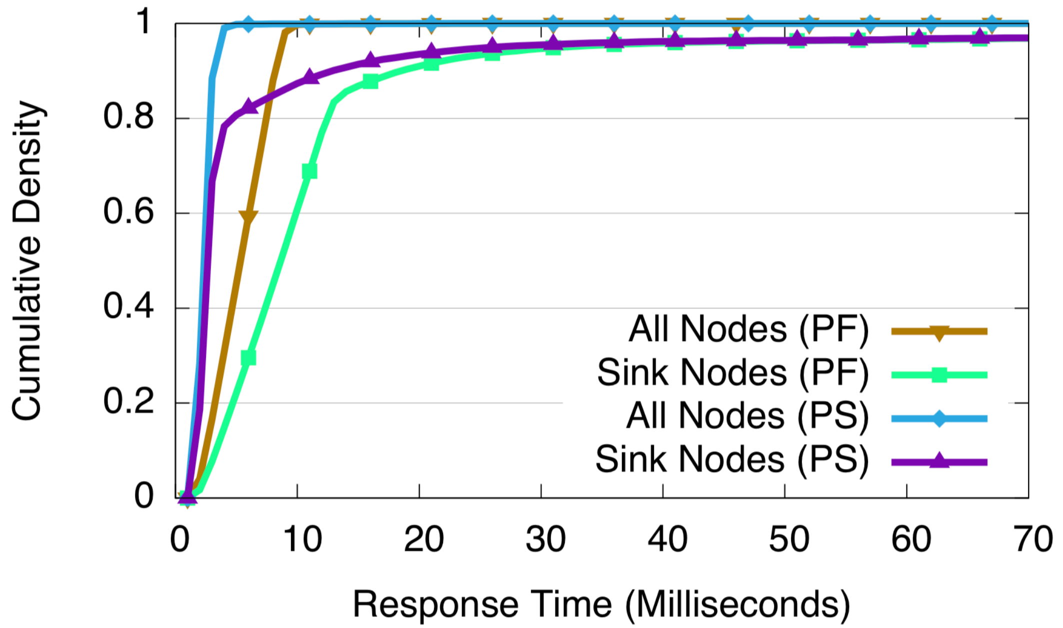
Evaluation

- Provenance storage growth & Query performance



ProvSave incurs less storage costs

Performance test on Neo4j



ProvThings can respond quickly
to real-time monitoring system

2. ContexIoT, *Jia et al. NDSS' 17*

Conclusion

- ProvThings is a framework for collection, management, and analysis of data provenance in IoT

- Limitations
  - Static Source Code Instrumentation
    - Unable to handle dynamic features of a language
  - Device Integrity
    - ProvThings assumes that the devices are not compromised
    - Compromised devices may cause wrong provenance graphs

2. ContexIoT, *Jia et al. NDSS' 17*

Questions?