FlowFence: Practical Data Protection for Emerging IoT Application Frameworks

Earlence Fernandes, Justin Paupore, Amir Rahmati, Daniel Simionato, Mauro Conti, Atul Prakash

University of Michigan, University of Padova

Published at USENIX Security 2016

sed on authors' slides

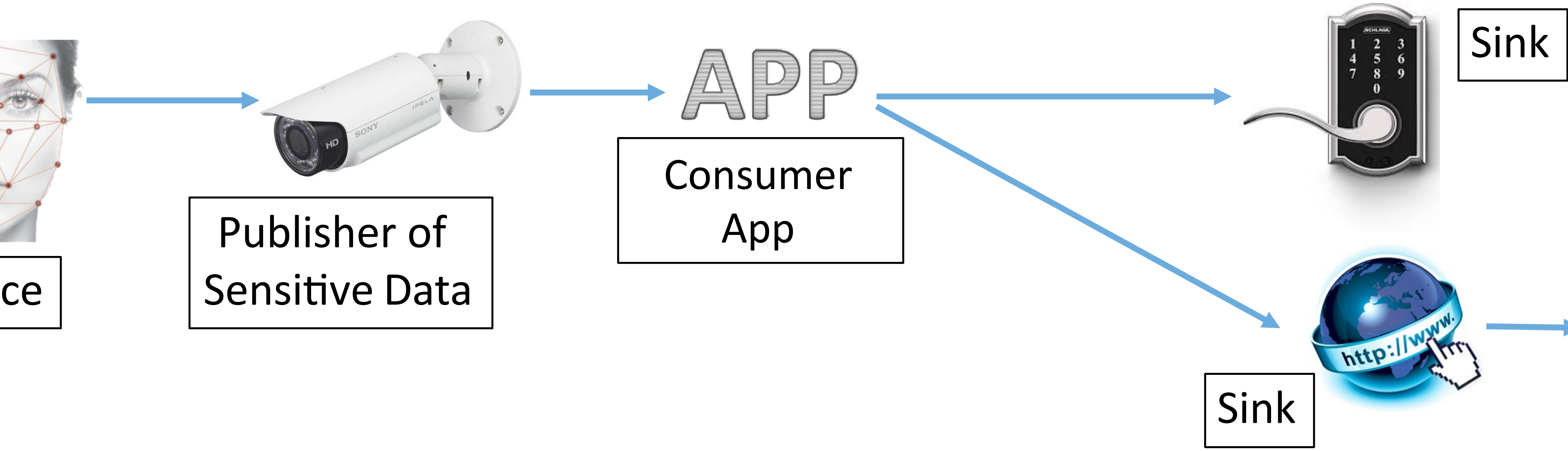
Presenter: Qi W

les/Quantified Self

Connected Healthcare

Smart Homes

Emerging IoT App Frameworks



nlock door if face is recognized
ome-owner can check activity
om Internet

- App needs to compute o
sensitive data to provide
useful service
- But has the potential to
leak data

Existing IoT frameworks have permission based access control

hings

Smart home API

[Smart Homes]

Google Fit API

[Wearables]

Android Sensor API

[Quantified Self]

e.g., capability.lockCodes in SmartThings (pincode
FITNESS_BODY_READ scope in Google Fit (heart r

- Permissions control what data an app can access
- Permissions do not control how apps use data, once they have access

FlowFence

Flow-control is a first-class primitive

based flow control
- mponent-level information tracking
- w enforcement through label policies

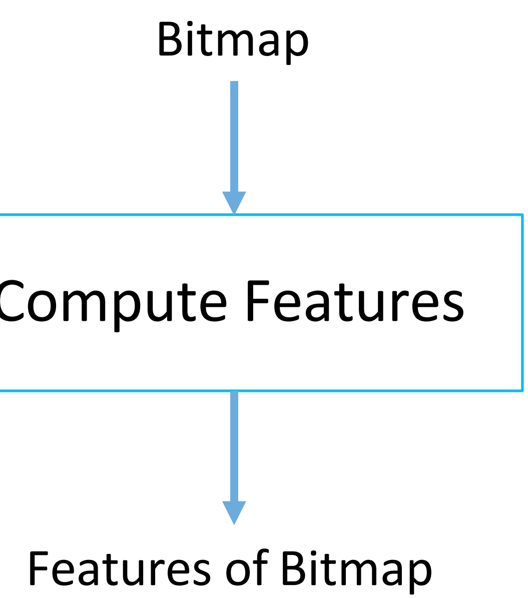
+

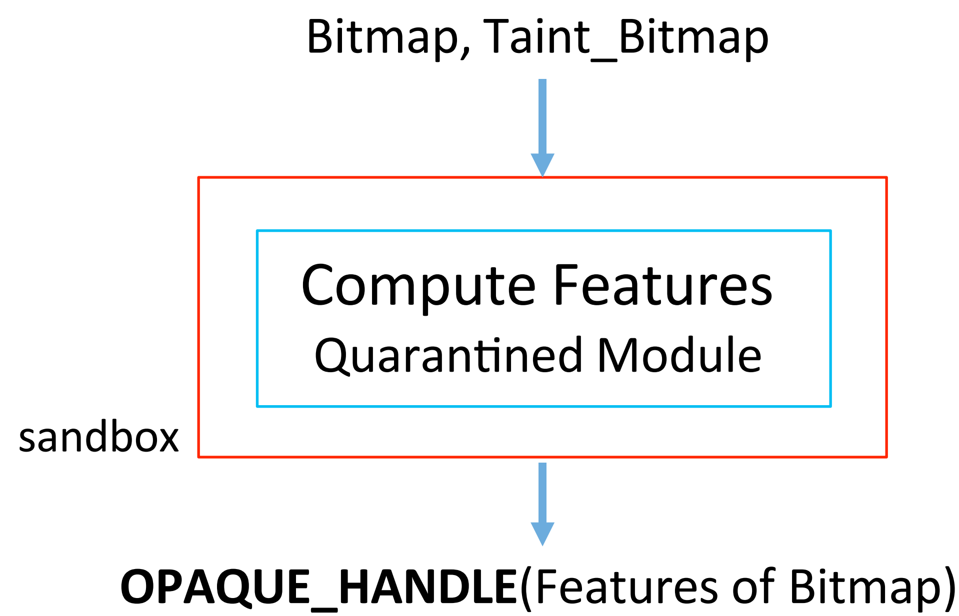
Language-based flow control
- Restructure apps to obey flow
- Developer declares flows

=

FlowFence
- Support of diverse publishers and consumers of data, with publisher and consumer flow policies
- Allows use of existing languages, tools, and OSes

Quarantined Modules and Opaque Handles

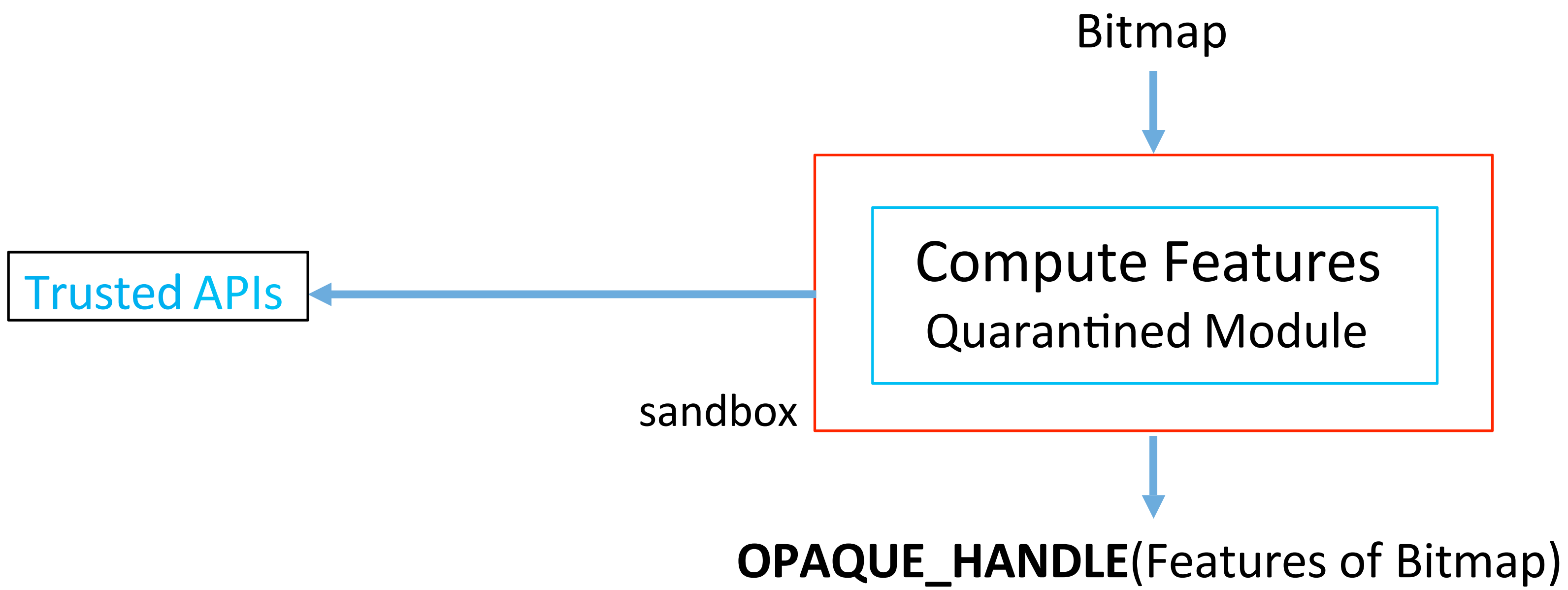




e computation runs with
e rights to access sensitive
tmap data

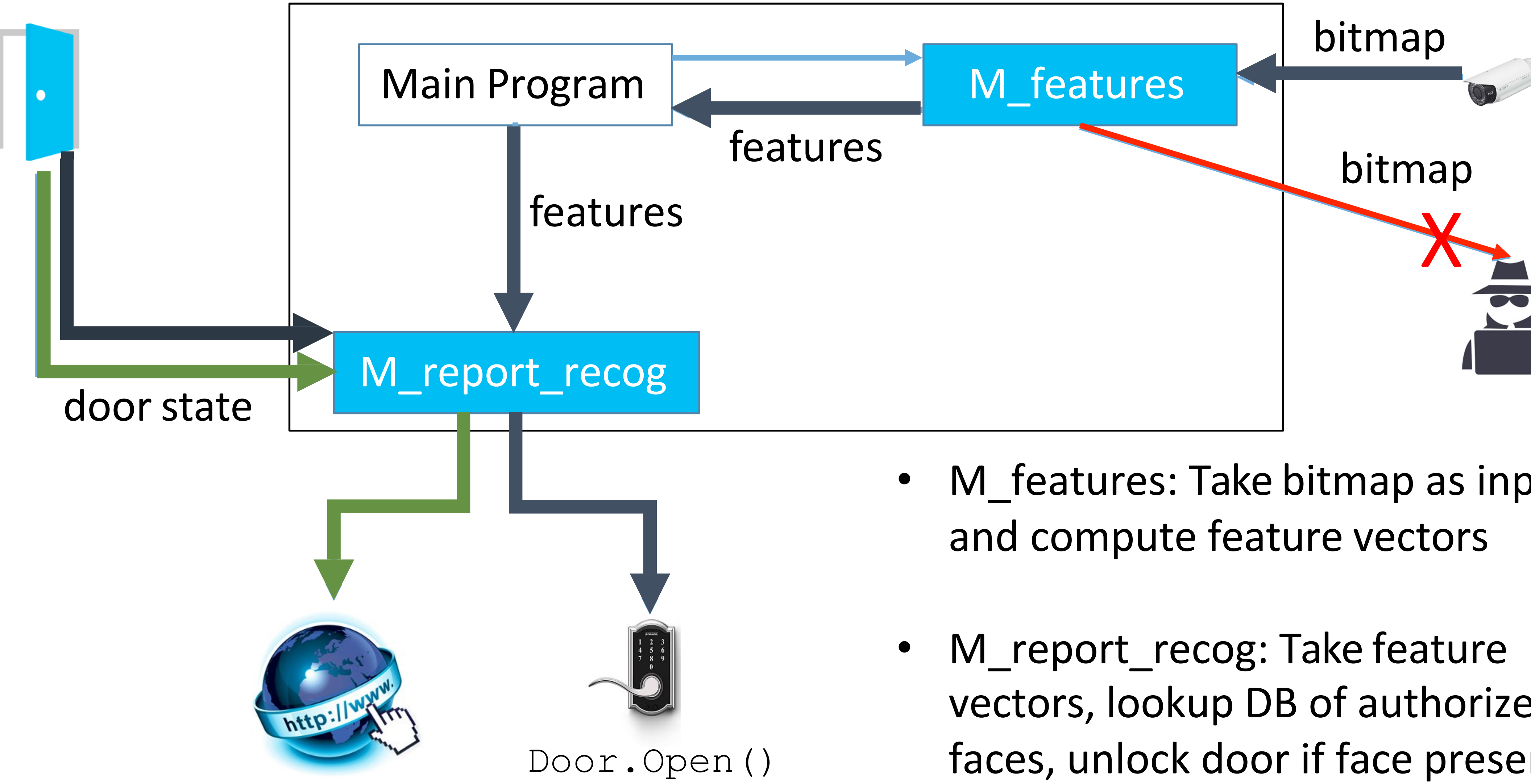
- Submit a computation that runs in a sand
- All sensitive data is available only in sand

Quarantined Modules and Opaque Handles



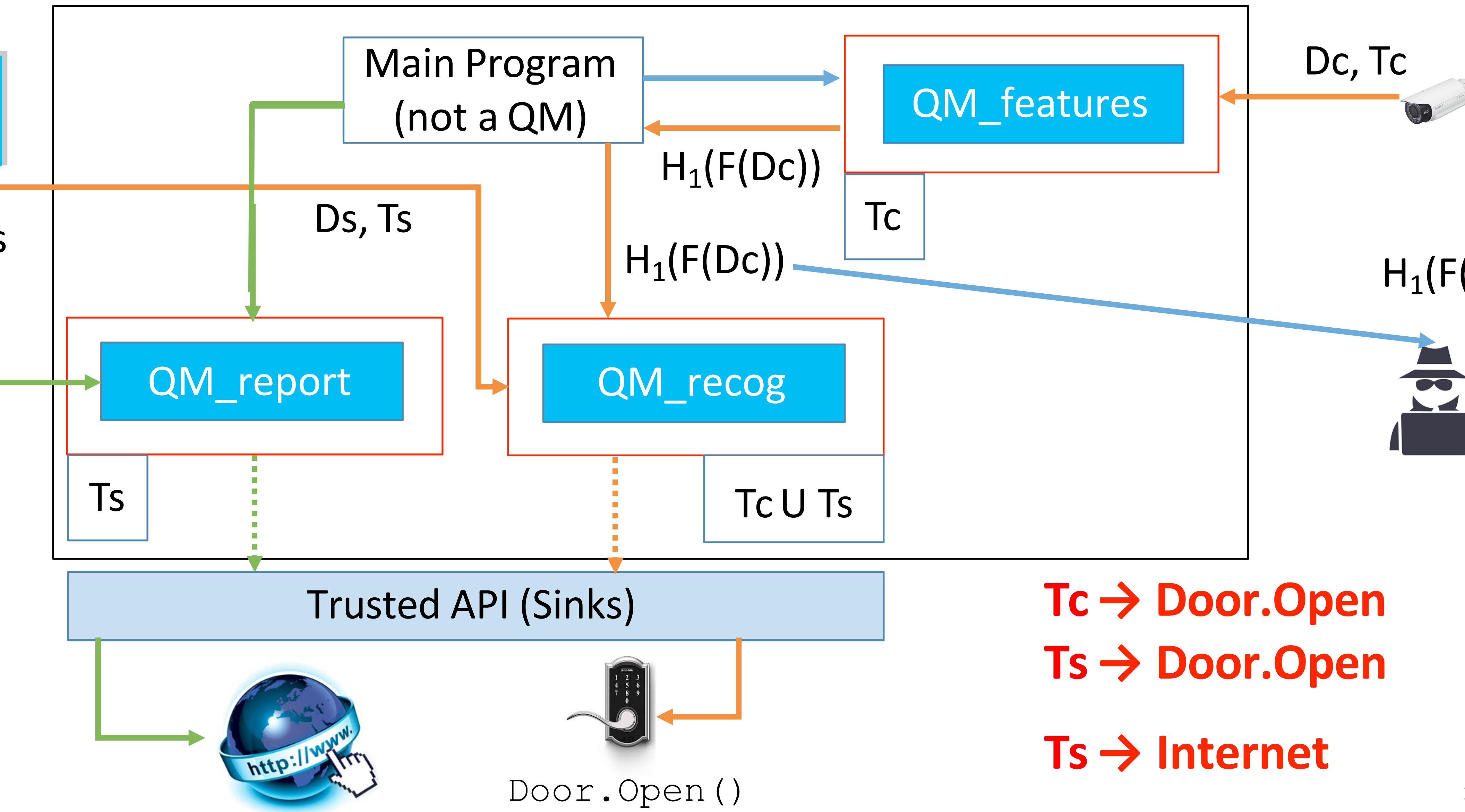
Quarantined Modules can also access FlowFence-provided Trusted APIs
Trusted APIs check the taint labels of the caller against a flow policy

Face Recognition App Example



- M_features: Take bitmap as inp
and compute feature vectors
- M_report_recog: Take feature
vectors, lookup DB of authorize
faces, unlock door if face prese
Report door state

FlowFence – Refactored App



Taint Labels and Flow Policies

H_1	{T1, T2, …}	F(D1)

App_ID	Label_Name
om.camera.publisher	Taint_Camera

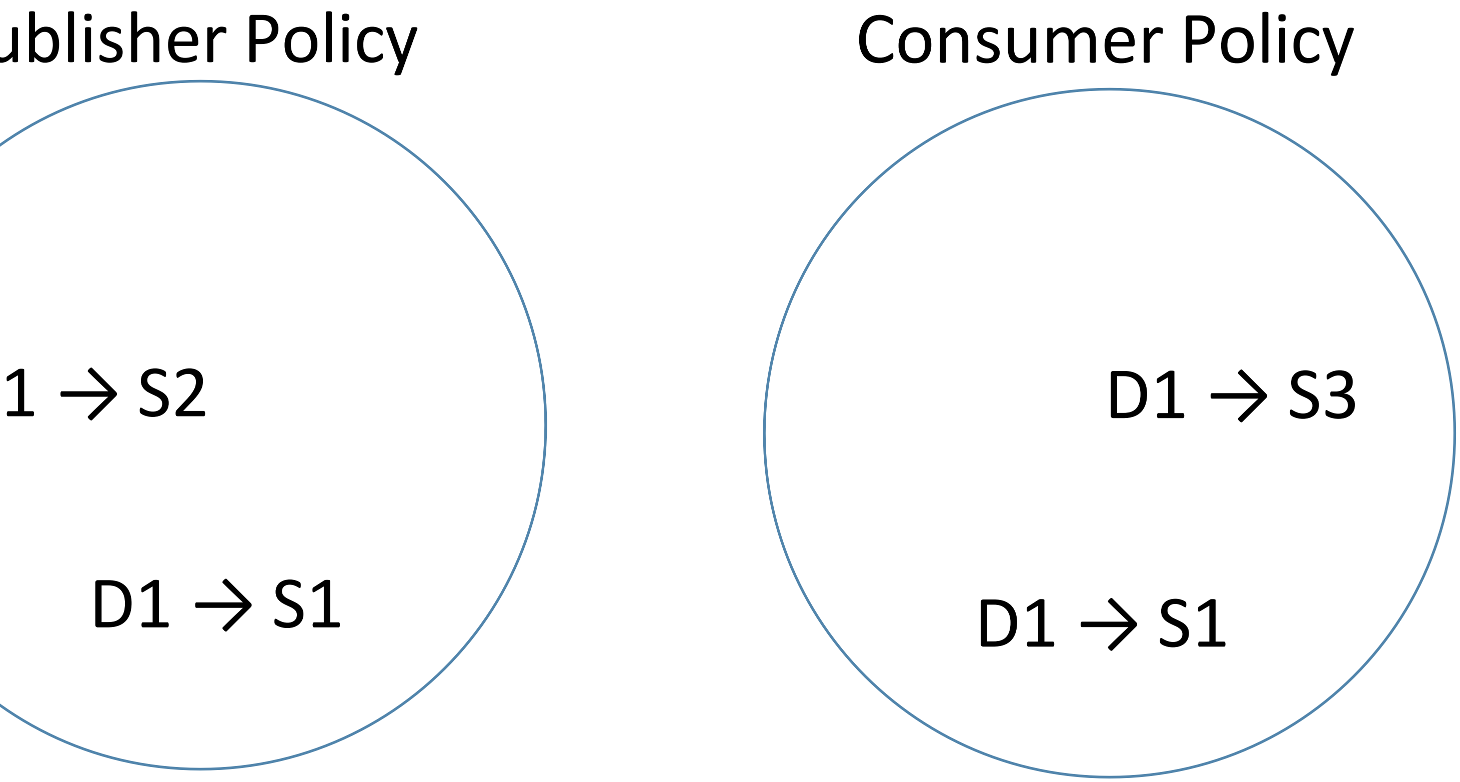
Example Policy

{
  Taint_Camera → UI,
  Taint_HeartR → Internet
}

App_ID – unique application identifier on the underlying OS

Label_Name – well-known string that identifies the type of da

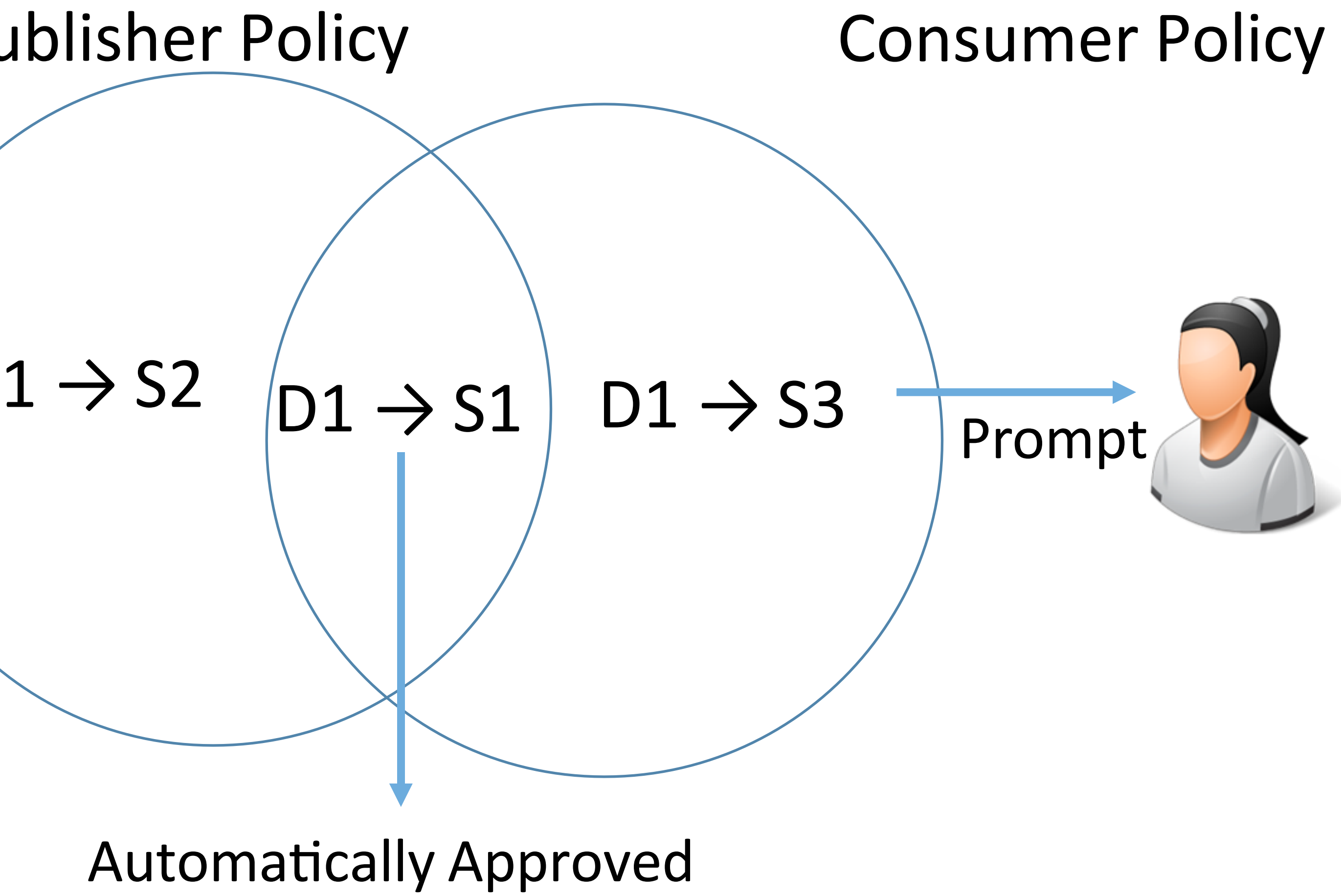
Publisher and Consumer Flow Policies



{ Publisher;
Taint_Camera → UI
}

{ Consumer;
Taint_Camera → Door.C
Taint_DoorState → Doo
Taint_DoorState → Inte
}

Publisher and Consumer Flow Policies



{ Publisher;
  Taint_Camera → UI
}

{ Consumer;
  Taint_Camera → Door.(
  Taint_DoorState → Doo
  Taint_DoorState → Inte
}

Data Sharing Mechanisms in Current IoT Frameworks

Smart home API

[Smart Homes]

Google Fit API

[Wearables]

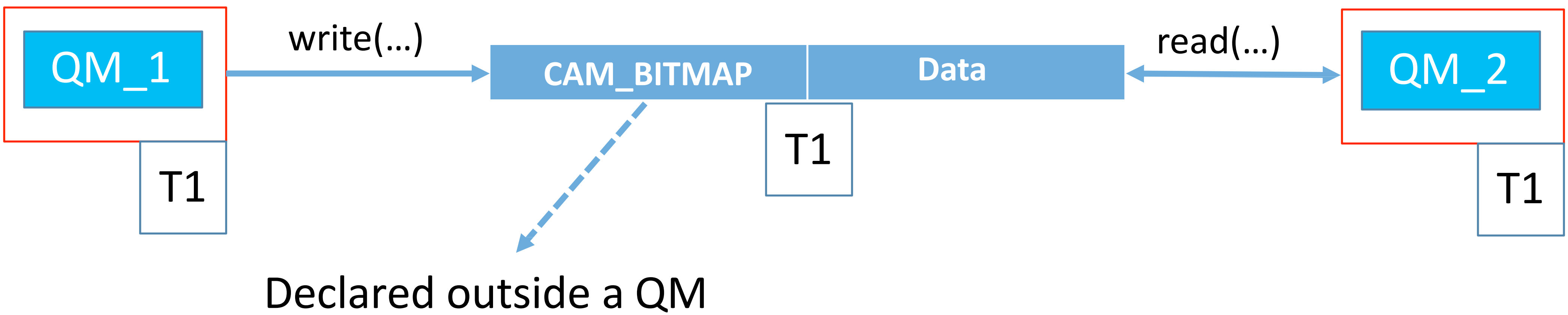
Android Sensor API

[Quantified Self]

- Polling Interface
  - App checks for new data
- Callback Interface
  - App is called when new data available
- Device Independence
  - E.g., many types of heart rate sensors produce “heart beat” data
  - Consumers should only need to specify “what” data they want, without specifying “how”

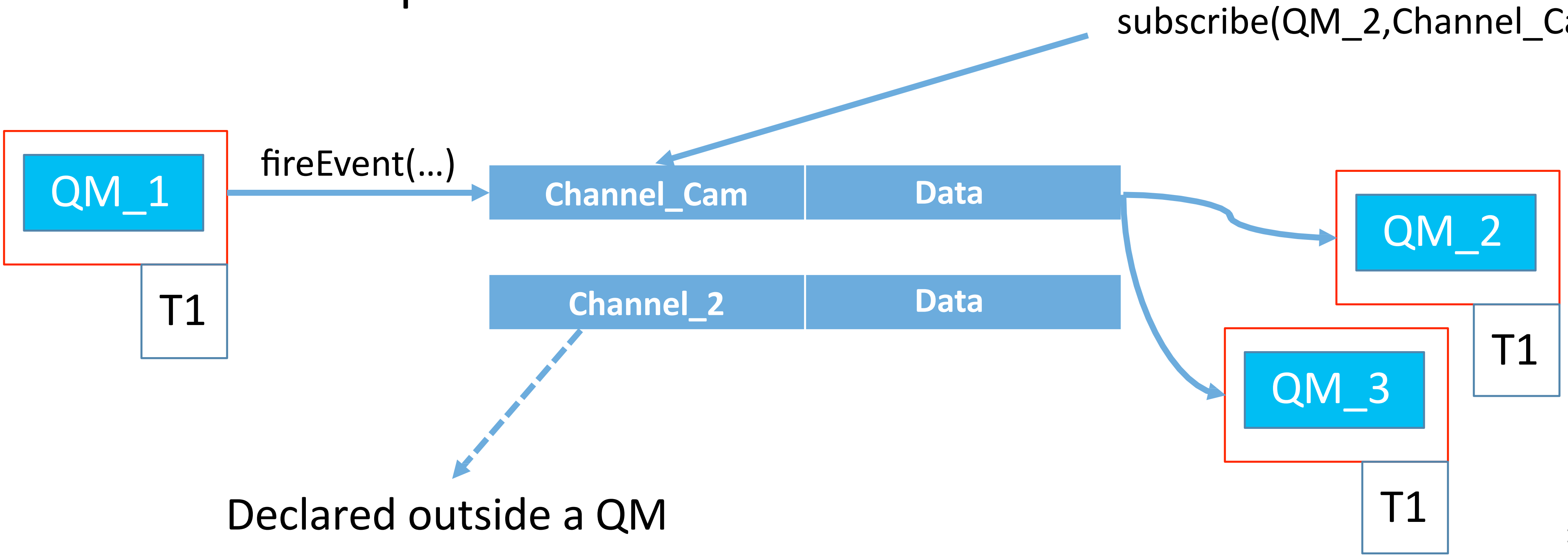
Key-Value Store –
Polling Interface/Device Independence

- ach app gets a single Key-Value Store
- n app can only write to its own Key-Value Store
- pps can read from any Key-Value Store
- eys are public information because consumers need to know about the



Event Channels –
Callback Interface/Device Independence

Apps can declare statically in code, their intended channels
Only the owner of a channel can fire an event
Channel name is public information



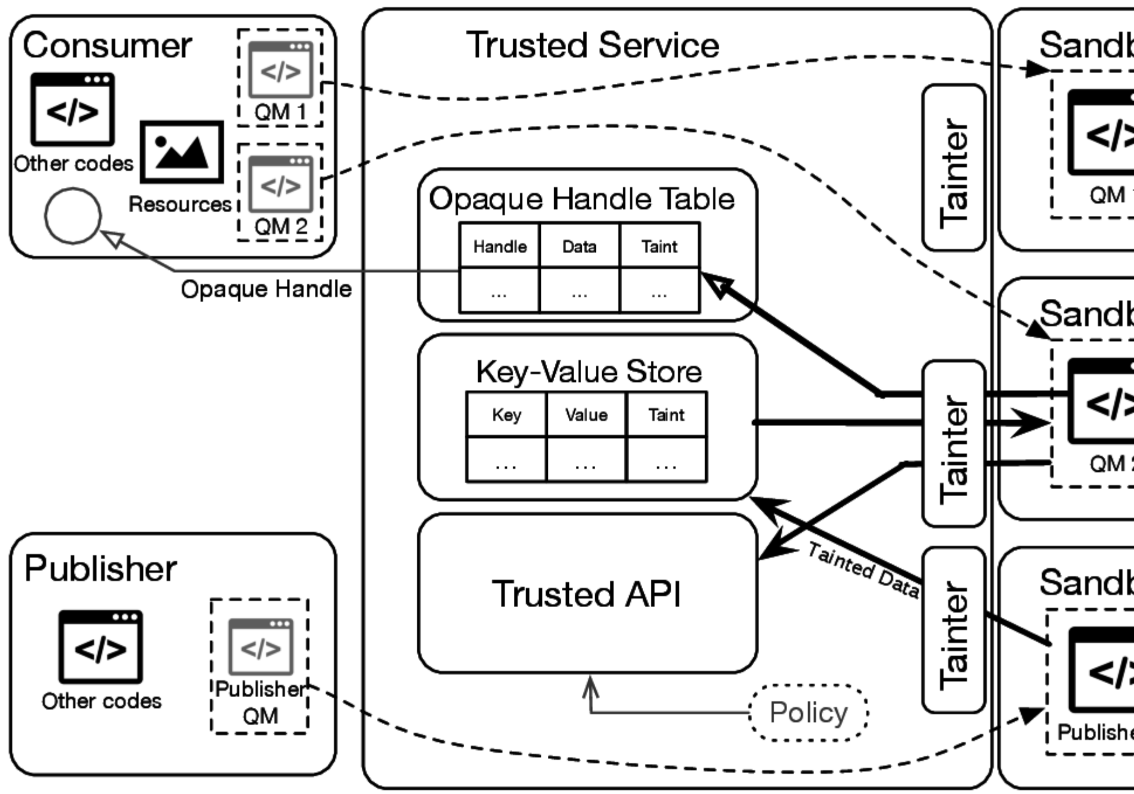
FlowFence Implementation

IoT Architectures

- Cloud
- Hub

”

- isolatedProcess = true for sandboxe
- Supports native code



Evaluation Overview

What is the overhead on a micro-level in terms of computation and memory?

Per-Sandbox Memory Overhead	2.7 MB
QM Call Latency	92 ms
Data Transfer b/w into Sandbox	31.5 MB/s

Comparable to IoT device ops over
area-network, e.g., Nest, SmartThi

Nest cam peak bandwidth is 1.2 M

Can FlowFence support real IoT apps securely?

Ported 3 Existing IoT Apps: SmartLights, FaceDoor, HeartRateMonitor	Required adding less than 140 lines per app; FlowFence isolates flows

What is the impact of FlowFence on macro-performance?

FaceDoor Recognition Latency	5% average increase
HeartRateMonitor Throughput	0.2 fps reduction on average
SmartLights end-to-end latency	+110 ms on average

Porting IoT Apps to FlowFence

App	Data Security Risk	Original LoC	FlowFence LoC	Flow Request
SmartLights	Can leak location information	118	193	Loc → Switch
FaceDoor	Can leak images of people	322	456	Cam → Lock, Doorstate → Lock Doorstate → Net
HeartRateMon	Can leak images and heart rate	257	346	Cam → UI

artLights, FaceDoor – 2 days of porting effort each, HeartMon – 1 day of porting ef

Macro-performance of Ported Apps

aceDoor Enroll Latency

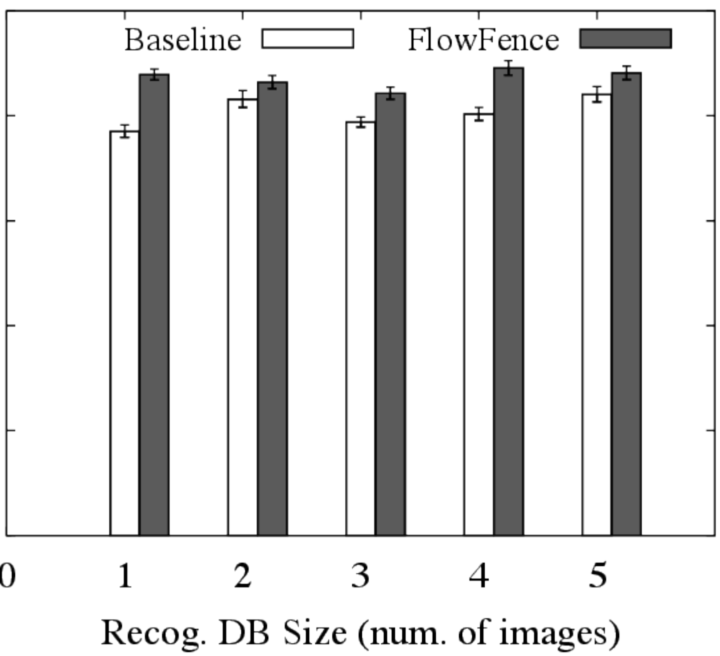
aseline	811 ms (SD = 37.1)
wFence	937 ms (SD = 60.4)

SmartLights End-To-End Latency

Baseline	160 ms (SD = 69.9)
FlowFence	270 ms (SD = 96.1)

eDoor Recognition Latency

(612x816 pixels)



HeartRateMon Throughput

Throughput w/o Image Processing	23.0 (SD=0.7) fps	22.9 (SD=0.
Throughput w/ Image Processing	22.9 (SD=0.7) fps	22.7 (SD=0.

Summary

merging IoT App Frameworks only support permission-based access contro
lalicious apps can steal sensitive data easily

owFence explicitly embeds control and data flows within app structure;
evelopers must split their apps into:

- Set of communicating Quarantined Modules with the unit of communication being Opaque Handles – taint tracked, opaque refs to data
- Non-sensitive code that orchestrates QM execution

owFence supports publisher and consumer flow policies that enable building
ecure IoT apps

/e ported 3 existing IoT apps in 5 days; Each app required adding < 140 LoC

lacro-performance tests on ported apps indicate FlowFence overhead is
easonable: e.g., 4.9% latency overhead to recog. a face & unlock a door

Discussion

- Vhat's the limitation of FlowFence?
- low is the usability of FlowFence to developers and users?
- low to improve FlowFence?
- Vhat makes protecting IoT challenging?
- s FlowFence able to mitigate the attacks we discussed in last class?

Instruction-Level Flow Analysis Techniques

Dynamic Taint Tracking

ne granularity
o developer effort
gh computational overhead
ay need special h/w for acceleration
plicit flows can leak information
nited OS/Language flexibility

IoT devices (and hubs) have
constrained hardware

OS and Language Diversity;
[Supports Rapid Developme

Static Taint Tracking

ne granularity
o developer effort
plicit flows can leak information
C and async. code can leak information

Fundamental Trigger-Action
Nature of IoT apps = Lots of
async. code