Securing Publish-Subscribe Systems with Scalable Fine-Grained Access Control

Spencer Henwood, Steve Willoughby, Jacob Olinger, and Primal Pappachan
Database and Internet Privacy (DIPr) Lab, Portland State University

Contact: henwood@pdx.edu

DIPr Lab

Portland State UNIVERSITY

Our Goal: Define and implement a scalable fine-grained access control model for Internet of Things (IoT) applications.

1. Motivation

Publish/subscribe architecture is popular for IoT systems, but performing access control for distributed networks of insecure devices is inefficient and doesn't scale well [1]. Modern IoT applications require expressive access control policies defining who can access what data, for what purposes, and under what circumstances.

Research Questions

- Can we create an end-to-end access control system that **stores**, **retrieves**, and **enforces** policies accurately in a pub/sub system?
- What is the performance impact of access control in pub/sub systems?

2. Background

Publish-Subscribe

Publish-subscribe (pub/sub) is a common networking architecture for IoT. There are two categories of clients: publishers that send data and subscribers that receive data. Messages are categorized by topic and routed through a broker. MQTT is a popular pub/sub protocol for IoT.

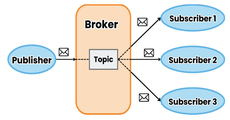

Figure 1. Publish-subscribe architecture diagram.

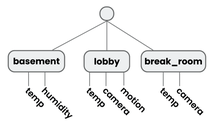

Figure 2. MQTT topic tree.

3. Policy Model

We developed an expressive policy model based on attribute-based access control (ABAC) [2].

Access Control Model

< top, subj, pur, obj, ctx, act >

top	Topic pattern
subj	Subject attribute expression
pur	Purpose of access request
obj	Object attribute expression
ctx	Context (e.g. time, location)
act	Action to take: GRANT or DENY

We then adapted that policy model into a relational database entity-relation diagram for policy storage and retrieval (Figure 3).

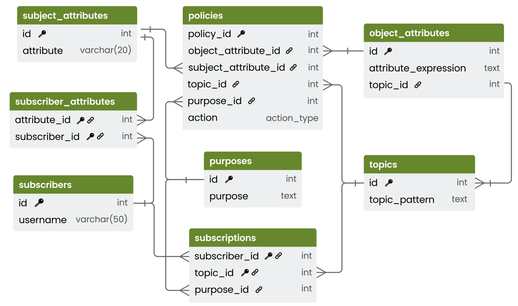

Figure 3. Policy database entity-relation diagram.

4. Enforcement Strategy

Policy enforcement can occur at two stages: at **subscribe-time** and at **publish-time.**

Subscribe-Time

Enforcement occurs when a client subscribes to a topic. It considers topic, subject attribute, and purpose. The steps are labeled in yellow in Figure 4.

1 - 3 The broker receives a subscribe message and passes it to the PEA.

4 - 7 The PEA queries for policies that apply to this subscription through the policy database interface.

8 The PEA evaluates the retrieved policies and returns a grant/deny decision to the broker.

Publish-Time

Enforcement occurs when a message is published. The steps are labeled in pink in Figure 4.

1 - 10 The broker receives a publish message and the PEA filters policies by applicable object attributes.

11 - 19 The broker sends the publish message to the subscriber and queries the filtered policies for matching policies.

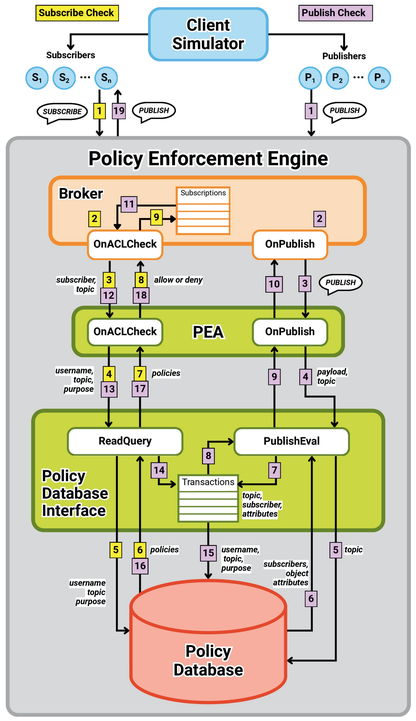

Figure 4. Policy enforcement architecture. The yellow numbers follow the subscribe-time policy check, and the pink numbers follow the publish-time policy check.

5. Access Control Environment

Built for this project:

- Policy Enforcement Agent (PEA)
- Policy Database Interface
- Policy Database, built in MySQL

Adapted from pre-existing modules:

- Mochi MQTT Broker
- Python MQTT Client Simulator [3]

6. Results

The environment was run for 20 minutes with and without the PEA performing subscribe-time enforcement. The time elapsed between a message being published and received was recorded. The experiment was run with 7 publishers, 5 subscribers, and 5 policies. Fewer messages were delivered with the PEA active, with increased latency for publish messages.

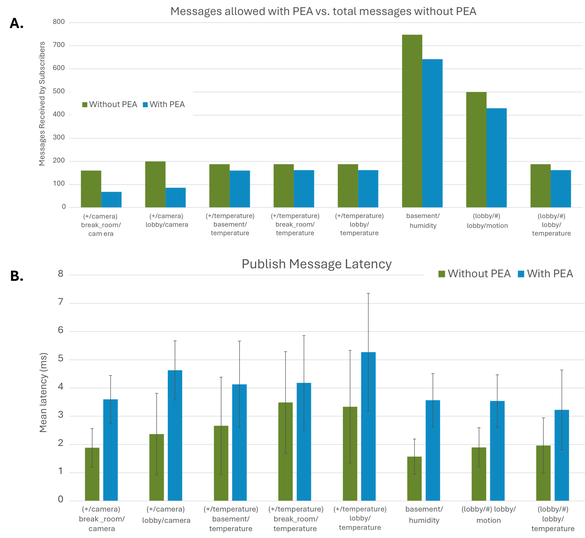

Figure 5. (A) Number of messages received with the PEA vs. without. (B) Latency of publish messages with the PEA vs. without.

7. Conclusions and Future Work

- We developed a fine-grained access control policy model for pub/sub systems and built a relational database to store and administer policies.
- We implemented a proof-of-concept that demonstrated the efficacy of enforcing policies.

Future Work

- Explore alternative enforcement strategies to reduce latency.
- Perform large-scale experiments to simulate realistic IoT scenarios.

References

[1] Ragothaman, K. et. al (2023). Access Control for IoT: A Survey of Existing Research, Dynamic Policies and Future Directions. Sensors, 23(4), 1805.
[2] Colombo, P. & Ferrari, E. (2018). Access Control Enforcement within MQTT-based Internet of Things Ecosystems. SACMAT 18.
[3] Damasceno, Rafael. Python MQTT Client Simulator. https://github.com/DamascenoRafael/mqtt-simulator