Prepared by: Dara Odukoya
Date: January 17, 2026
System Assessed: GrandMarina Hydrologic MQTT Pipeline (Development Environment)

Part 1: Reconnaissance

Traffic Interception

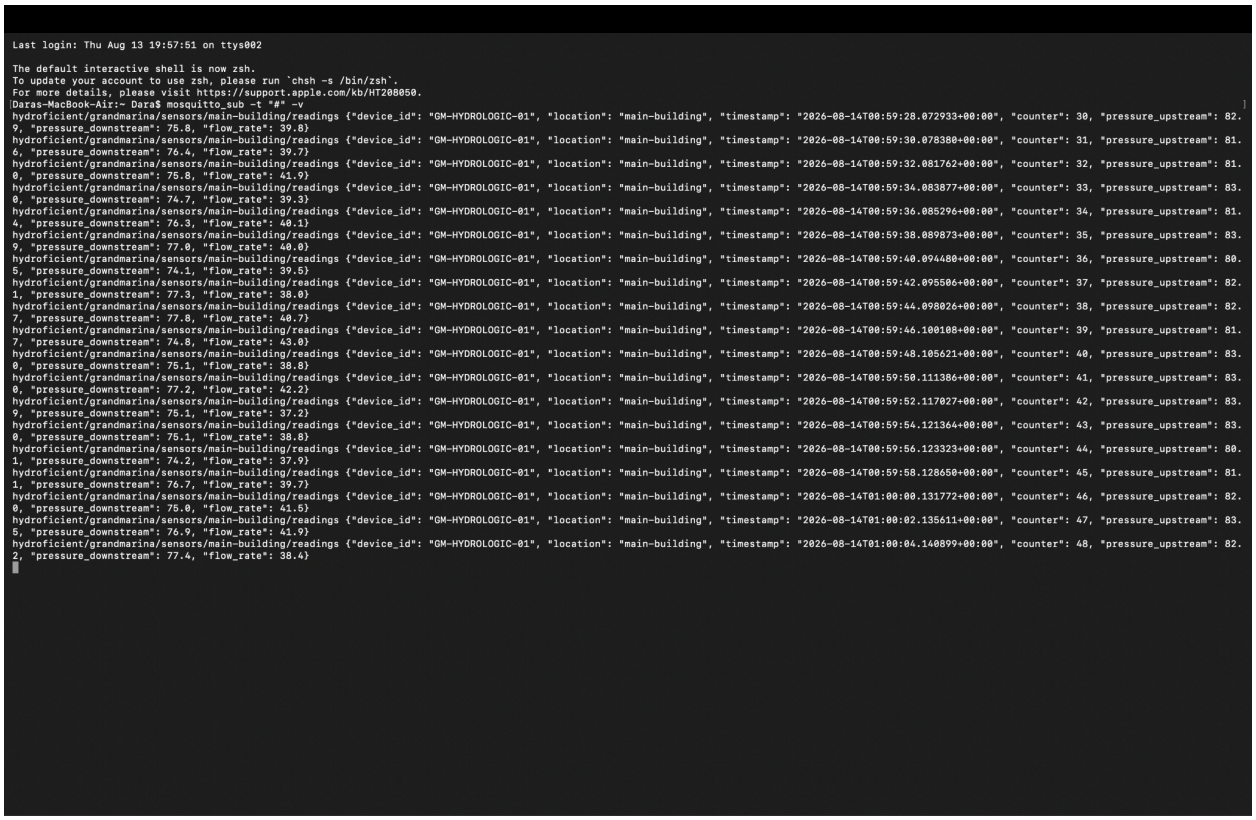
I ran my pipeline with three terminals (broker, publisher, subscriber), then opened a fourth terminal and ran:

```
Shell
mosquitto_sub -t "#" -v
```

Results after 30 seconds of capture:

- **Messages captured:** 17 messages
- **Topics observed:**
  - hydroficient/grandmarina/sensors/main-building/readings
  - medvault/riverside-general/devices/ward-a/readings
  - medvault/riverside-general/devices/ward-b/readings
- **Information visible in each message:**
  - Device ID (e.g., GM-HYDROLOGIC-01)
  - Location (e.g., main-building)
  - Timestamp (full ISO format with timezone, e.g 2026-08-14T00:59:28.072933+00:00)
  - Counter (sequence number)
  - Pressure upstream (e.g., 82.9)
  - Pressure downstream (e.g., 75.8)
  - Flow rate (e.g., 39.8)

Screenshot of Intercepted Traffic



Part 2: Vulnerability Analysis

Vulnerability	What's the Risk?	Evidence from My Pipeline	Potential Attack
No encryption	Data is transmitted in plain text, so anyone with network access can read every message as it passes by	My mosquitto_sub -t "#" -v capture showed complete, readable JSON messages, device ID, location, timestamp, pressure upstream/downstream, and flow rate, with no encoding or encryption at all	An attacker on the guest WiFi could learn normal pressure/flow ranges for main-building, then craft a convincing fake reading that blends in with real traffic instead of standing out as obviously fake

No authentication	Anyone who can reach the broker can connect to it, there's no login, password, or certificate required to join	I ran mosquitto_sub -t "#" -v from a fourth terminal with no credentials of any kind and immediately started receiving live data from GM-HYDROLOGIC-01	An attacker could connect a laptop to the same network and run the exact same command I did, no password guessing or exploit needed, just a standard MQTT client
No authorization	Once connected, there's nothing stopping a client from subscribing to or publishing on any topic, not just the ones relevant to their role	Subscribing to the wildcard # topic returned everything, sensor readings for main-building, with no restriction based on who I was or what I should be allowed to see	Someone without a legitimate reason to monitor the Kitchen/Laundry line could subscribe to it anyway, or worse, publish directly to commands/main-building/emergency/shutoff even though they have no operator role
No message verification	Messages have no signature or integrity check, so there's no way to confirm a message actually came from the real device or hasn't been altered in transit	The JSON payloads in my capture (device ID, counter, pressure/flow values) had no signature, hash, or authentication field, just plain values that any client could reproduce and publish themselves	An attacker could capture a real reading, replay it later to mask a leak, or publish a fabricated message under GM-HYDROLOGIC-01's topic and the dashboard would have no way to tell it apart from a legitimate one

Part 3: Remediation Recommendations

1. What defenses would I add?

1. **TLS Encryption (Port 8883)**
    - Encrypt all traffic between HYDROLOGIC devices, the cloud broker, and the dashboard
    - Prevents eavesdropping and man-in-the-middle attacks, directly addresses what my mosquitto_sub -t "#" -v capture exposed

    - Standard approach: configure Mosquitto to require TLS connections instead of the default unencrypted port
2. **Client Authentication**
    - Require username/password or client certificates for every device and dashboard connection
    - Only authorized HYDROLOGIC devices and operators can connect, my test showed I connected with zero credentials
    - Revoke credentials immediately if a device is compromised or a laptop/operator account is lost
3. **Topic-Based Access Control Lists (ACLs)**
    - Define rules like: "Device GM-HYDROLOGIC-01 can only publish to sensors/main-building/readings"
    - Dashboards can subscribe to readings but shouldn't be able to publish directly to device command topics without going through proper authorization
    - Prevents unauthorized publishing and limits the blast radius if one device or account is compromised
4. **Message Validation**
    - Check timestamps, reject messages older than a set window (e.g. 30-60 seconds)
    - Track counters, reject duplicate or out-of-sequence messages, this directly closes the replay attack scenario already identified (captured shutoff command replayed later)
    - Consider message signatures (HMAC) so the dashboard can verify a message actually came from the real device

2. What's most urgent to fix?

Priority Ranking:

Rank	Vulnerability	Why This Priority
1	**No authentication**	This is the front door. My test proved anyone can connect to the broker with no credentials at all, without fixing this, none of the other controls matter, since an attacker just walks past them.
2	**No encryption**	Once we control who connects, we need to protect what they send. Right now pressure, flow, and location data travel in plain text, exactly what my capture showed.
3	**No authorization**	With authentication and encryption in place, we need to limit what authenticated clients can actually do, so a compromised or misused account can't touch every device and every topic.

4	**No message verification**	Defense in depth. Even with the above controls, validating messages catches compromised devices or replayed commands, which is what stops the replay attack scenario specifically.

3. What trade-offs might these defenses have?

Defense	Trade-off
TLS Encryption	**Performance:** adds some CPU overhead and slight latency. With readings published every 5 seconds per device, this is unlikely to be a real problem at Grand Marina's scale (3 devices), but worth testing after implementation.
Client Authentication	**Complexity:** each of the 3 devices plus every operator account needs credentials or certificates. Small scale compared to the MedVault example (50 devices), but still needs a real provisioning process, not ad hoc passwords.
Access Control Lists	**Maintenance:** rules need updating any time a device is added or an operator's role changes. Misconfigured ACLs can silently block legitimate traffic or leave gaps, so this needs testing before going live.
Message Validation	**Clock Sync:** Timestamp validation requires devices to have synchronized clocks. Clock drift could cause valid readings to get rejected, so devices need NTP configured.