Section 1: Executive Summary

The Grand Marina Hotel relies on the HYDROLOGIC water management system to control water pressure and flow across its three main service lines: the main building, the pool/spa wing, and the kitchen/laundry wing. These devices monitor pressure and flow in real time, send that data to a dashboard, and can remotely open or close gates and trigger emergency shutoffs when something goes wrong. This system is projected to save the hotel 20% on water costs, but it also means a piece of critical infrastructure, water delivery for 500 rooms, 12 restaurants, and a full-service spa, now depends on connected devices talking to each other over the network.

This threat model looks at the security risks that could affect guest safety, hotel operations, and the water savings Marcus is counting on. Smart building and IoT systems are an increasingly common target for attackers, since a single compromised device can give access to an entire connected network, not just the one device itself.

Our analysis identified three critical concerns requiring immediate attention:

- Weak or default credentials on the system (like the MQTT broker) could let an attacker on the guest WiFi gain full access, not just to one device, but to all three water lines at once
- An attacker could flood the system with traffic and knock the dashboard offline, meaning staff would lose visibility into water pressure and flow right when they need it most
- Emergency shutoff commands could be captured and replayed later, potentially shutting off water to a wing of the hotel during a normal, busy day instead of an actual emergency

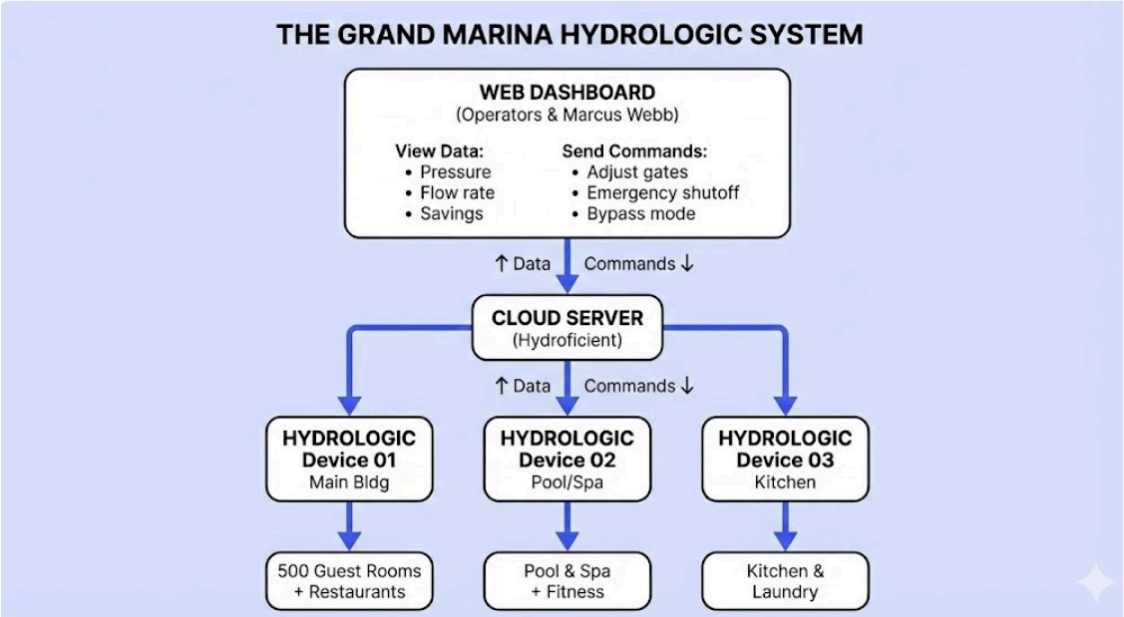
We recommend prioritizing strong, unique credentials and access controls for the MQTT broker first, since unauthorized access is the single scenario that could compromise all three devices simultaneously and has the widest possible impact on the hotel.

Section 2: System Overview

The Grand Marina's HYDROLOGIC Deployment:

- 500-room luxury hotel (Hydroficient customer)
- 3 HYDROLOGIC flow management devices, one per water service line (Main Building, Pool/Spa, Kitchen/Laundry)
- Cloud-based monitoring and control (Hydroficient cloud server)
- Web dashboard for operators and Marcus Webb
- Remote gate control and emergency shutoff capability, plus a bypass mode

Data flow diagram:



How It Works:

1. Each HYDROLOGIC device measures pressure and flow rate on its water line in real time
2. Devices send that data up to the Hydroficient cloud server
3. The cloud server processes the data and forwards it to the web dashboard
4. Operators and Marcus Webb view pressure, flow rate, and savings on the dashboard
5. If needed, operators send commands back down through the cloud server, like adjusting gates, triggering emergency shutoff, or switching to bypass mode
6. The cloud server relays those commands to the correct device, which acts on them immediately

Section 3: Asset Inventory

List the key assets and their CIA priorities (use your work from Step 2):

Asset	Description	C Priority	I Priority	A Priority
HYDROLOGIC Devices	3 flow management units	Critical	Critical	Low

Asset	Description	C Priority	I Priority	A Priority
Web Dashboard	Operator monitoring interface	Critical	Medium	Low
Cloud API	Device-to-cloud communication	Critical	Critical	High
Remote Controls	Gate adjustments, emergency shutoff	Critical	Critical	Critical
Consumption Data	Savings reports, billing records	Medium	Critical	Medium

Section 4: STRIDE Analysis

This is the core of your threat model. For **each major component**, analyze all six STRIDE categories:

Component 1: HYDROLOGIC Devices

Threat	Scenario	Likelihood	Impact	Risk
Spoofing	Attacker joins guest WiFi, connects to the MQTT broker, and publishes fake pressure readings under Device 01's known topic name, tricking the dashboard into showing normal pressure when there's actually a leak	Medium	High	High
Tampering	Attacker performs a MITM on the device-to-cloud connection (MQTT traffic isn't encrypted) and rewrites the flow rate values in transit before they reach the cloud	Medium	High	High
Repudiation	Device commands aren't signed or tied to a specific sender ID, so if an attacker sends a fake gate command, there's no log proving it wasn't a legitimate device	Medium	Medium	Medium

Info Disclosure	Attacker sniffs guest WiFi traffic with a tool like Wireshark and reads raw MQTT pressure/flow data in plaintext, since MQTT commonly runs unencrypted on port 1883	High	Medium	High
Denial of Service	Attacker floods the MQTT broker with junk publish messages from a script, so Device 01's real readings get lost or delayed in the noise	Medium	Critical	High
Elevation of Privilege	Attacker finds the device still has its factory default admin login on its local config interface and uses it to change device settings directly	Medium	Critical	Critical

Component 2: Web Dashboard

Threat	Scenario	Likelihood	Impact	Risk
Spoofing	Attacker brute forces or guesses a weak operator password (e.g. "hydro2024") and logs into the dashboard as a real operator	Medium	High	High
Tampering	Attacker with dashboard access edits displayed savings numbers or suppresses a real pressure alert so staff don't notice a problem	Low	High	Medium
Repudiation	All operators share one login, so if someone opens a gate they shouldn't have, there's no way to tell which person actually did it	High	Medium	High
Info Disclosure	Dashboard session cookie is sent over guest WiFi without HTTPS, so an attacker sniffing traffic captures the cookie and hijacks the session without needing a password	Medium	High	High
Denial of Service	Attacker runs a credential-stuffing bot against the login page, hammering it with requests until the dashboard becomes too slow or unresponsive for real operators	Low	Medium	Low
Elevation of Privilege	A read-only viewer account exploits a bug where the "send command" button is only hidden in the UI, not blocked on the backend,	Low	Critical	Medium

	letting them fire gate commands they shouldn't be able to			

Component 3: Cloud API

Threat	Scenario	Likelihood	Impact	Risk
Spoofing	Attacker gets ahold of a leaked or hardcoded API key (e.g. pulled from a device's firmware) and uses it to send requests that look like they're coming from a real HYDROLOGIC device	Medium	High	High
Tampering	Attacker intercepts an API call sending a "close gate 10%" command and changes the payload to "close gate 100%" before it reaches the device	Low	Critical	High
Repudiation	API doesn't log which device certificate or IP a request actually came from, so a malicious command can't be traced back to its source	Medium	Medium	Medium
Info Disclosure	A misconfigured endpoint doesn't properly scope requests by customer, so a request meant to pull Grand Marina's consumption data accidentally returns data for other Hydroficient customers too	Low	High	Medium
Denial of Service	Attacker scripts a flood of API requests that exceeds what the server can handle, so real device data and dashboard requests start timing out	Medium	High	High
Elevation of Privilege	Attacker changes the customer ID or device ID parameter in an API request (an IDOR-style flaw) and gets back control over a device that isn't tied to their account	Low	Critical	Medium

Component 4: Remote Controls (Gate/Shutoff)

Threat	Scenario	Likelihood	Impact	Risk
Spoofing	Attacker on the network sends a fake "open gate" command directly to a device, pretending to be the cloud server, because the device doesn't verify who the command is actually coming from	Medium	Critical	Critical
Tampering	Attacker captures a legitimate shutoff command in transit and changes the target device field from Device 03 (kitchen) to Device 01 (main building) before it's delivered	Low	Critical	High
Repudiation	Commands aren't digitally signed, so an attacker-issued shutoff looks identical to a real one in any logs, making it impossible to prove who triggered it	Medium	High	High
Info Disclosure	Attacker monitors MQTT traffic over time and learns the exact command structure/syntax used for gate and shutoff commands, giving them what they need to craft their own	High	Medium	High
Denial of Service	Attacker rapidly fires conflicting gate commands (open, close, open, close) at a device, causing the valve motor to jam or fail from being overworked	Low	High	Medium
Elevation of Privilege	An attacker with only dashboard viewer access finds the direct API endpoint the gate button calls and hits it themselves, bypassing the UI restriction that was supposed to keep them read-only	Low	Critical	Medium

Section 5: Risk Summary

Critical Risks:

1. **Default admin credentials on HYDROLOGIC devices (Elevation of Privilege)** — Devices still have factory default logins on their local config interface. Anyone who finds this can take direct control of a device, bypassing the cloud and dashboard entirely. This is the same weakness behind your "Unauthorized Access" scenario, ranked your #1 most dangerous attack.
2. **Unauthenticated remote commands (Spoofing)** — Devices don't verify that a gate/shutoff command actually came from the cloud server. An attacker on the network can send a fake "open gate" or "shutoff" command directly, and the device will execute it. Given these controls affect water delivery to guest rooms, the kitchen, and the pool, a successful spoof here has hotel-wide, potentially safety-related impact.

High Risks:

1. **Unencrypted MQTT traffic (Information Disclosure)** — Pressure and flow data travels in plaintext, commonly over port 1883. Anyone on guest WiFi with a packet sniffer can read it, and over time can learn the exact command structure well enough to forge their own.
2. **Man-in-the-middle tampering on device data (Tampering)** — Because traffic isn't encrypted, an attacker can alter pressure/flow readings in transit, this is the same weakness that lets the "replayed shutoff command" scenario from your attack mapping work.
3. **Weak/shared dashboard credentials (Spoofing)** — Passwords are guessable and shared across operators rather than individual, so one compromised login grants full dashboard access with no way to trace who's actually behind an action.
4. **No session encryption on the dashboard (Info Disclosure)** — Session cookies travel over guest WiFi without HTTPS, so an attacker can hijack an active operator session without ever needing a password.
5. **Flood attacks on the broker and API (Denial of Service)** — Both the MQTT broker and cloud API can be overwhelmed with junk traffic, delaying or hiding real sensor data and blocking legitimate dashboard/API requests, this is your "Denial of Service" scenario, ranked #2 most dangerous.
6. **Leaked or hardcoded API keys (Spoofing)** — If a key is pulled from device firmware, an attacker can send commands to the cloud API that look like they're coming from a real device.
7. **No command authentication trail (Repudiation)** — Gate and shutoff commands aren't signed or tied to a verified sender, so a malicious command is indistinguishable from a real one in the logs, this makes any incident much harder to investigate after the fact.

Medium Risks:

1. Shared logins on the dashboard create audit gaps (Repudiation) — no way to tell which operator performed a given action.

2. UI-only access restrictions on the dashboard (Elevation of Privilege) — a "view only" account can potentially call restricted commands directly if the backend doesn't enforce the same restriction as the UI.
3. Missing customer-scoping on some API endpoints (Info Disclosure) — risk of one customer's data being returned in another customer's request.
4. Parameter manipulation on the API (Elevation of Privilege) — changing a device/customer ID in a request could grant control over a device that isn't yours.
5. Command flooding causing physical wear (Denial of Service) — rapid conflicting gate commands could jam or damage valve motors over time.

Section 6: Recommended Mitigations

For Critical Risks

Risk	Proposed Mitigation	Implementation Complexity
Default admin credentials on HYDROLOGIC devices	Force unique, strong credentials on every device during setup; disable factory default logins entirely	Low — configuration change
Unauthenticated remote commands	Require devices to verify a signed command (e.g. HMAC or device certificate) actually came from the legitimate cloud server before executing it	High — firmware/ architecture change

For High Risks

Risk	Proposed Mitigation	Implementation Complexity
Unencrypted MQTT traffic	Move to MQTT over TLS (port 8883) for all device-to-cloud communication	Medium — firmware update required
Man-in-the-middle tampering	Same TLS fix as above, plus	Medium — firmware update

on device data	message-level integrity checks so tampered payloads are rejected even if intercepted	required
Weak/shared dashboard credentials	Enable MFA and require individual operator logins instead of one shared account	Low — configuration change
No session encryption on the dashboard	Enforce HTTPS across the dashboard, including secure, HttpOnly session cookies	Low — configuration change
Flood attacks on the broker and API	Add rate limiting and basic anomaly detection on both the MQTT broker and cloud API	Medium — cloud configuration
Leaked or hardcoded API keys	Move away from static hardcoded keys; issue rotating, per-device credentials or short-lived tokens	Medium — architecture change
No command authentication trail	Log every command with a verified device/operator identity attached, and digitally sign commands so they can't be forged or repudiated	Medium — application update