# IoT MQTT Security Research Platform - Project Documentation

## 1.3. Research Objectives

This project aims to design and implement an open-source, rule-based detection and monitoring system for the MQTT protocol to be deployed alongside brokers. By finding and using public MQTT datasets as a base, the group can identify the important fields for creating detection rules against popular MQTT attacks (such as DDoS or flood attack), and simulate traffic to be as close to their real-life counterparts as possible for testing the detection system.

### Project Components Delivered:

**Data Processing Pipeline:**

* `build\_canonical\_dataset.py`: Normalizes heterogeneous MQTT/IoT CSV exports into a canonical schema
* `feature\_extract.py`: Extracts machine learning features from canonical datasets
* 9 standardized IoT device datasets (Temperature, Humidity, CO2, Motion, etc.)
* Canonical dataset output with 60-70% field standardization across disparate sources

**IoT Traffic Simulation:**

* `unified\_simulator.py`: Comprehensive MQTT simulator supporting both enhanced realistic IoT payloads and legacy CSV replay modes
* Support for 9 different IoT device types with realistic telemetry generation
* Configurable broker connections, publish intervals, and device scaling
* Production-ready containerized deployment via Docker

**MQTT Attack Simulation Suite (9 Attack Scripts):**

1. `script\_flood.py`: Message flooding attacks

2. `script\_wildcard.py`: Wildcard subscription abuse

3. `script\_bruteforce.py`: Topic brute-force attacks

4. `script\_payload\_anomaly.py`: Payload anomaly injection

5. `script\_retain\_qos.py`: Retain/QoS abuse attacks

6. `script\_topic\_enumeration.py`: Topic enumeration attacks

7. `script\_duplicate\_id.py`: Duplicate client ID conflicts

8. `script\_reconnect.py`: Reconnection storm attacks

9. `script\_qos2\_abuse.py`: QoS 2 exactly-once delivery abuse

**Monitoring and Detection Infrastructure:**

* EMQX MQTT broker integration via Docker Compose
* Real-time traffic monitoring capabilities
* Unified attack demonstration interface (`demo\_all\_attacks.py`)
* Comprehensive logging and CSV output for analysis

### Performance Targets:

The system is designed to handle monitoring for approximately 300 MQTT clients with the following performance objectives:

* **True Positive Rate (TPR)**: ≥ 90%
* **False Positive Rate (FPR)**: ≤ 5%
* **Processing Latency (p95)**: ≤ 300ms
* **Additional CPU Usage**: ≤ 30% above idle state
* **Additional RAM Usage**: ≤ 400MB above idle state

## 1.4. Significance of the Study

### SME Practical Implications

This research provides a lightweight, interpretable, and reproducible MQTT attack monitoring and detection system specifically designed for small- and medium-scale enterprise (SME) IoT environments in which resources are tight and administrators may not have advanced knowledge about security. While intrusion detection systems or advanced machine learning plans put a burden on resources and are frequently pricey to install and maintain, the rule-based approach enables real-time recognition of typical MQTT abuses (e.g., publish flooding, wildcard abuses, brute-force authentication, client takeover, and payload anomalies), thereby reducing investigation time and maximizing response effectiveness.

**Key SME Benefits:**

* **Low Resource Requirements**: Production-ready code optimized for minimal overhead
* **Easy Deployment**: Complete Docker Compose stack for one-command deployment
* **Interpretable Results**: Rule-based detection provides clear, actionable insights
* **Cost-Effective**: Open-source solution with no licensing fees
* **Scalable Architecture**: Supports 300+ concurrent devices with defined performance metrics

### Portability and Deployment Ease

The entire monitoring–detection pipeline (Rule Engine → EMQX → Fluent Bit/Telegraf → InfluxDB → Grafana) is packaged as a containerized application using Docker Compose, thus simplifying deployment, scaling, and redeployment across different environments (pilot projects, laboratories, production). This significantly reduces time between deployability and research prototypes.

**Deployment Features:**

* **Containerized Architecture**: Complete Docker Compose stack with EMQX broker
* **Environment Flexibility**: Works across development, testing, and production environments
* **Simplified Management**: Single configuration file for entire monitoring pipeline
* **Version Control**: Git-ready project structure with comprehensive documentation

### Standardization of MQTT Logging

Through conceptualization of one common schema across disparate datasets (~60–70% overlap of fields standardized), our work reduces analysis errors, facilitates uniform queries, and ensures experimental reproducibility. This standardised log format also offers prerequisites for future extensions to machine learning and cross-dataset comparisons without difficulty.

**Standardization Achievements:**

* **Canonical Schema**: 20 standardized fields covering core MQTT protocol elements
* **Multi-Dataset Support**: Successfully processes 9 different IoT device types
* **Field Mapping**: Automatic detection and mapping of 60+ field name variants
* **Data Quality**: Comprehensive payload sanitization and format validation
* **Export Compatibility**: CSV output compatible with ML frameworks and analysis tools

### Quantifiable Goals for System Performance

The project defines clear quality metrics for near-real-time detection for a system of 300 devices: True Positive Rate (TPR) ≥ 90%, False Positive Rate (FPR) ≤ 5%, 95th percentile latency ≤ 300 ms, additional CPU usage ≤ 30%, and additional RAM usage ≤ 400 MB above idle. These quantifiable metrics both promote practical measurability for SMEs and comparability for subsequent studies.

**Performance Measurement Framework:**

* **Benchmark Suite**: 9 comprehensive attack scenarios for performance validation
* **Metrics Collection**: Real-time performance monitoring with CSV logging
* **Load Testing**: Support for 300+ concurrent device simulation
* **Resource Monitoring**: CPU and memory usage tracking during attack scenarios
* **Latency Analysis**: p95 latency measurement for detection response times

### Scholarly and Practical Contribution

The research offers (i) technically valid rule set for MQTT attacks, (ii) reproducible monitoring pipeline in real-time, and (iii) systematic evaluation framework (scenarios, ground-truthing, TPR/FPR/latency metrics). These contributions individually and together integrate scholarly work with practical IoT security needs and lay foundational ground for future ML-based experimentation.

**Technical Contributions:**

1. **Attack Taxonomy**: Comprehensive classification of 9 MQTT attack patterns

2. **Detection Rules**: Rule-based engine capable of identifying malicious MQTT behavior

3. **Simulation Framework**: Realistic IoT traffic generation for security testing

4. **Evaluation Methodology**: Systematic approach to measuring detection system performance

5. **Open Source Implementation**: Complete codebase available for research reproducibility

**Research Impact:**

* **Academic Value**: Peer-reviewable methodology and quantifiable results
* **Industrial Relevance**: Direct applicability to SME IoT security challenges
* **Future Research**: Foundation for machine learning-based MQTT security systems
* **Community Benefit**: Open-source contribution to IoT security research community

## Project Architecture Overview

### System Components

IoT MQTT Security Research Platform  
├── Data Processing & Normalization  
│ ├── CSV Dataset Ingestion (9 IoT device types)  
│ ├── Canonical Schema Mapping  
│ └── Feature Extraction for ML  
├── Traffic Simulation  
│ ├── Realistic IoT Device Simulation  
│ ├── Historical Data Replay  
│ └── Configurable Traffic Patterns  
├── Attack Generation  
│ ├── 9 Distinct MQTT Attack Types  
│ ├── Configurable Attack Parameters  
│ └── Real-time Attack Orchestration  
├── Monitoring & Detection  
│ ├── EMQX MQTT Broker Integration  
│ ├── Real-time Traffic Analysis  
│ └── Rule-based Detection Engine  
└── Performance Evaluation  
 ├── Metrics Collection (TPR/FPR/Latency)  
 ├── Resource Usage Monitoring  
 └── Comprehensive Reporting

### Deployment Architecture

The system utilizes a containerized microservices architecture:

* **EMQX MQTT Broker**: Handles MQTT protocol communications
* **IoT Simulator**: Generates realistic device traffic and attack patterns
* **Detection Engine**: Processes MQTT messages and applies security rules
* **Monitoring Dashboard**: Provides real-time visibility into system performance

This architecture ensures scalability, maintainability, and ease of deployment across different environments while meeting the specified performance requirements for SME IoT security monitoring.

## Figure 1. MQTT Protocol-enabled IoT Architecture

┌─────────────────────────────────────────────────────────────────┐  
│ IoT Device Layer │  
├─────────────────────────────────────────────────────────────────┤  
│ Temperature │ Humidity │ Motion │ CO2 │ Light │ Smoke │ etc... │  
│ Sensors │ Sensors │Sensors │Sens.│Sensors│Sensors│ │  
└──────┬───────┴────┬─────┴───┬────┴─────┴───┬───┴───┬───┴─────────┘  
 │ │ │ │ │  
 └────────────┼─────────┼──────────────┼───────┘  
 │ │ │  
 ┌───▼─────────▼──────────────▼───┐  
 │ MQTT Broker │  
 │ (EMQX 5.0.13) │  
 │ │  
 │ Topics: site/tenant/zone/ │  
 │ device/telemetry │  
 └─────────────┬──────────────────┘  
 │  
 ┌─────────────────────────▼─────────────────────────┐  
 │ Security Monitoring Layer │  
 ├─────────────────────────────────────────────────┤  
 │ Attack Detection Engine │ Performance Monitor │  
 │ ┌─────────────────────────┐│ ┌─────────────────┐│  
 │ │ • Flood Detection ││ │ • TPR ≥ 90% ││  
 │ │ • Wildcard Abuse ││ │ • FPR ≤ 5% ││  
 │ │ • Brute Force ││ │ • Latency ≤300ms││  
 │ │ • Payload Anomaly ││ │ • CPU ≤ +30% ││  
 │ │ • QoS Abuse ││ │ • RAM ≤ +400MB ││  
 │ │ • Topic Enumeration ││ └─────────────────┘│  
 │ │ • Client ID Conflict ││ │  
 │ │ • Reconnect Storm ││ │  
 │ │ • QoS2 Abuse ││ │  
 │ └─────────────────────────┘│ │  
 └─────────────────────────────────────────────────┘  
 │  
 ┌─────────────────────────▼─────────────────────────┐  
 │ Analytics Layer │  
 ├─────────────────────────────────────────────────┤  
 │ Data Pipeline │ Visualization │  
 │ ┌─────────────────────┐ │ ┌─────────────────────┐│  
 │ │ • CSV Normalization │ │ │ • Real-time Dashboard││  
 │ │ • Feature Extraction│ │ │ • Attack Metrics ││  
 │ │ • Schema Mapping │ │ │ • Performance Graphs││  
 │ │ • Data Validation │ │ │ • Alert Management ││  
 │ └─────────────────────┘ │ └─────────────────────┘│  
 └─────────────────────────────────────────────────┘

This architecture demonstrates the complete MQTT security monitoring pipeline, from IoT device simulation through attack detection to performance analysis, specifically designed for SME environments with quantifiable performance targets.