# Hướng dẫn Demo Dự án IoT MQTT

## 1. Dự án này làm gì?

* Mô phỏng 9 thiết bị IoT gửi telemetry qua MQTT dựa trên dữ liệu thật (MQTTset).
* Chuẩn hóa dữ liệu từ nhiều bộ (MQTTset, Edge-IIoTset, DoS/DDoS, Gotham) vào một **schema chuẩn** dùng cho phân tích/tạo luật phát hiện tấn công.
* Hạ tầng sẵn có: script Python, Docker stack (EMQX + simulator), pipeline xử lý dữ liệu.

Bạn có thể mở rộng lên 300 thiết bị (theo Final Idea) bằng cách nhân bản client và tích hợp thêm dataset.

## 2. Chuẩn bị trước khi demo

1. **Tải mã nguồn** vào máy (đảm bảo folder Do An IA).
2. **Đặt dữ liệu CSV** vào thư mục datasets/ (đã có sẵn, .gitignore bỏ qua). Nếu mới clone, hãy copy các file \*MQTTset.csv vào datasets/.
3. **Cài đặt Python 3.11** (nếu chưa có).
4. **Cài Docker Desktop** (nếu muốn chạy Docker demo).
5. Mở PowerShell/Terminal, chuyển tới thư mục dự án:

* cd "C:\Users\Admin\Desktop\Do An IA"

## 3. Thiết lập môi trường Python

1. Tạo và kích hoạt môi trường ảo (khuyến nghị):

* python -m venv .venv  
  .venv\Scripts\Activate

1. Cài phụ thuộc:

* pip install -r requirements.txt
* (Có paho-mqtt và pandas, cần cho simulator + xử lý dữ liệu.)

### Kiểm tra nhanh

* Sau khi cài xong, chạy pip list xem có paho-mqtt, pandas.

## 4. Xử lý dữ liệu (chuẩn hóa → đặc trưng)

### 4.1. Chuẩn hóa về schema chung

* Lệnh:
* python build\_canonical\_dataset.py --pattern "\*MQTTset.csv" --output canonical\_dataset.csv --chunksize 50000 --force
* Script sẽ đọc tất cả file CSV trong datasets/, map các cột đồng nghĩa (timestamp, client\_id, topic, qos, retain, payload\_length, packet\_type, protocol, Label, …), lọc giao thức IoT, ghi ra canonical\_dataset.csv.
* Kết quả: ~2.55 triệu bản ghi cho 9 thiết bị mẫu.

### 4.2. Trích xuất đặc trưng

* Lệnh:
* python feature\_extract.py canonical\_dataset.csv --out features\_canonical\_dataset.csv
* Sinh các trường: timestamp, client\_id, topic, qos, retain\_flag, dup\_flag, payload\_length, iat\_sec, Label, … – phục vụ rule-based detection.

### 4.3. Minh chứng

* Mở 5 dòng đầu của file chuẩn hóa:
* python - <<"PY"  
  import pandas as pd  
  print(pd.read\_csv("canonical\_dataset.csv", nrows=5))  
  PY
* Cho giảng viên thấy các trường chuẩn đã gom gọn.

## 5. Mô phỏng 9 thiết bị MQTT

### 5.1. Chạy bằng Python (không Docker)

python simulator\_from\_csv.py --broker localhost --port 1883 --publish-interval 0.2

* Script đọc từng file CSV từ datasets/, decode payload hex → JSON, publish lên topic site/tenantA/home/<device>/telemetry.
* Tham số --publish-interval giúp cố định tốc độ gửi, có thể thay đổi.
* Muốn dừng nhấn Ctrl + C.

### 5.2. Chạy bằng Docker (có EMQX)

1. Khởi động stack:

* docker-compose up --build -d
* (Dockerfile đã copy folder datasets/, simulator tự chạy.)

1. Kiểm tra container:

* docker ps --format "table {{.Names}}\t{{.Status}}\t{{.Ports}}"

1. Xem log simulator (muốn chắc chắn 9 thiết bị đã kết nối):

* docker logs --tail 10 mqtt-simulator

1. Mở dashboard EMQX: http://localhost:18083 (đăng nhập mặc định admin/public).
2. Dừng stack sau demo: docker-compose down (thêm --volumes nếu muốn xóa volume tạm).

5.3. Kịch bản tấn công MQTT (theo playbook)

- Chuẩn bị: đảm bảo broker EMQX/Mosquitto đang chạy (docker-compose up emqx) và có thể bật script simulator để tạo baseline hợp lệ.

- Các script tấn công nằm tại thư mục gốc dự án, log đính kèm được append vào file CSV để dùng cho báo cáo/rule.

Script Flood – Publish Flood

- Lệnh ví dụ: python script\_flood.py --broker localhost --clients 50 --msg-rate 200 --topic-template "factory/{client}/telemetry" --log-csv flood.csv --duration 60

- Phát sinh 50 client attacker, mỗi client publish ~200 msg/s với payload 128B; CSV flood.csv lưu timestamp/client/topic/mid/result.

- Theo dõi trên Grafana panel "Publish/sec by client" hoặc EMQX Dashboard → Connections để thấy spike và áp dụng rule PublishFlood.

Script Wildcard – Wildcard Abuse

- Lệnh ví dụ: python script\_wildcard.py --broker localhost --client-id attacker-wildcard --topics "#" "$SYS/#" "factory/+/+/#" --resubscribe-interval 10 --log-csv wildcard.csv --duration 180

- Script giữ kết nối nghe lén các wildcard rộng, tự resubscribe 10s/lần; log wildcard.csv lưu SUBACK và message nhận được.

- Kiểm tra rule wildcard\_sub\_count, so sánh role người dùng và log EMQX $events/session\_subscribed để chứng minh hành vi lạm dụng.

Script Brute-force – Subscribe Enumeration

- Chuẩn bị danh sách topic: tạo file targets.txt (mỗi dòng 1 topic) hoặc dùng tham số --topic-template/--topic-count.

- Lệnh ví dụ: python script\_bruteforce.py --broker localhost --topics-file targets.txt --rate 20 --rotate-every 100 --log-csv brute.csv

- Script gửi tuần tự yêu cầu SUBSCRIBE, xoay client-id sau mỗi 100 topic; log ghi trạng thái granted/denied và latency để thống kê fail\_count.

- Dựa trên log brute.csv và query InfluxDB để chứng minh distinct\_topics\_1min > ngưỡng và tỷ lệ SUBACK lỗi, kích hoạt rule BruteForceSubscribe.

## 6. Trình bày với giảng viên

### 6.1. Flow gợi ý (khoảng 10 phút)

1. **Giới thiệu ngắn (1 phút):** mục tiêu đề tài, tại sao cần mô phỏng.
2. **Dữ liệu & chuẩn hóa (3 phút):**
   * Cho xem schema chuẩn và lệnh build\_canonical\_dataset.py.
   * Hiển thị canonical\_dataset.csv/features\_canonical\_dataset.csv.
3. **Mô phỏng thực tế (3 phút):**
   * Mô tả simulator → EMQX.
   * Mở log docker logs --tail 10 mqtt-simulator (hoặc chạy script Python) để chứng minh 9 thiết bị hoạt động.
4. **Liên hệ với Final Idea (2 phút):**
   * Giải thích 9 loại thiết bị hiện tại là mẫu; plan nhân bản thành 300 thiết bị (theo bảng phân bổ 5 khu).
5. **Kế hoạch tiếp theo (1 phút):**
   * Nối vào Edge-IIoTset, DoS/Gotham.
   * Xây rule engine (InfluxDB/Grafana).

### 6.2. Visual gợi ý

* Có thể vẽ sơ đồ đơn giản: Device Simulator → EMQX → (Tương lai: Telegraf/InfluxDB/Grafana).
* In bảng phân bố 300 thiết bị trong “Final Idea” để giải thích vì sao có trùng lặp tên thiết bị.

## 7. FAQs (Câu hỏi thường gặp)

| Câu hỏi | Trả lời gợi ý |
| --- | --- |
| **Vì sao chỉ có 9 thiết bị mô phỏng?** | Hiện có dữ liệu thật cho 9 loại. Khi cần 300 thiết bị (theo Final Idea), ta nhân bản dataset bằng cách đổi client\_id/topic theo từng khu. |
| **Tích hợp Edge-IIoTset/DoS/Gotham thế nào?** | Đặt file vào datasets/, chạy lại build\_canonical\_dataset.py (có alias cho device\_id, topicfilter, …). Có thể chỉnh --pattern, --protocols. |
| **Phát hiện tấn công ra sao?** | Dựa vào schema chuẩn + features\_canonical\_dataset.csv: ví dụ burst attack = đếm số PUBLISH/s theo client\_id; brute-force = đếm CONNECT fail với connack\_code; retain abuse = Lọc retain\_flag=1. |
| **Khó khăn lớn nhất?** | Dataset lớn (GB), nhiều cột trùng/khác tên, payload hex. Đã giải quyết bằng xử lý chunk, bảng alias, hàm sanitize\_text. |
| **Mục tiêu cuối cùng?** | Xây “Trung tâm điều khiển” IoT: simulator → EMQX → Telegraf/InfluxDB → Rule Engine/Grafana Alerts. |