# Report

> Mục tiêu của đồ án: xây một pipeline **TLS IDS** (IDS cho traffic TLS) theo kiểu "sensor bắt lưu lượng → trích xuất fingerprint/feature → ML phát hiện bất thường → backend lưu + cảnh báo → (tuỳ chọn) auto-block trên firewall".

---

# Module 1

## Suricata (Sensor sniffing + sinh log eve.json)

- **SPAN port (Switch)**: cấu hình SPAN ở chế độ *Both (Bidirectional)* để mirror đủ 2 chiều (ClientHello + ServerHello) → Suricata mới trích đủ fingerprint JA3/JA3S (và metadata TLS). filecite☆turn7file2☆L4-L10
- **Card mạng Sensor phải bật Promiscuous** (nhận mọi frame, kể cả không gửi cho MAC của nó):
  - `ip link set <tên_card_monitoring> promisc on` filecite☆turn7file2☆L5-L6
  - Nếu không bật, NIC sẽ drop frame "không phải của mình" → Suricata "mù" dù đã cắm đúng dây SPAN. filecite☆turn7file2☆L5-L6
- **Khuyến nghị 2 NIC vật lý**:
  - NIC 1 (Management): có IP để SSH, chạy Backend/Frontend.
  - NIC 2 (Monitoring): không cần IP, chỉ dùng để cắm SPAN + promisc. filecite☆turn7file2☆L8-L10
- **Chạy Suricata trong container ở host network**: Suricata cần "thấy" interface thật của máy sensor (NIC monitoring). Vì vậy dùng `network_mode: host` là hợp lý (bridge network thường không sniff được NIC thật). filecite☆turn7file2☆L7-L7
- **Dữ liệu Suricata tạo ra**:
  - Suricata parse packet → phát hiện handshake TLS → ghi log theo JSON-line vào `eve.json` (mỗi dòng là 1 event).
  - Log được mount ra host / volume để module Python đọc realtime (xem Module 2).

**Luồng dữ liệu (Module 1)**

```
Switch SPAN (mirror traffic)
  → NIC monitoring (promisc)
    → Suricata (host network)
      → /var/log/suricata/eve.json (JSON lines)
```

---

# Module 2

## python-real-time-service (tail eve.json → feature extraction → ML → gọi API backend)

### Vai trò

- "Đọc log realtime" từ `eve.json` (log TLS do Suricata sinh ra), chỉ lọc `event_type=tls`.
- Trích xuất feature TLS/JA3/JA3S (cipher suites, groups, version enum, tỉ lệ weak/pfs, rule flags…).
- Chạy model ML (AutoEncoder + tùy chọn IsolationForest) → ra `anomaly` + score.
- **Chỉ gửi event "bất thường"** lên backend để lưu DB + tạo alert (giảm noise, giảm tải).

### Đầu vào

- File log: `.../suricata/eve.json` (JSON-lines).
- Mỗi dòng (1 event) có các field như:
  `timestamp, flow_id, src_ip, dest_ip, tls.version, tls.ja3.hash, tls.ja3s.string, tls.sni, ...`

### Xử lý realtime

- Pattern chính là "tail -f": chờ file tồn tại → seek end → loop đọc line mới → parse JSON → nếu là tls event thì xử lý.
- Sau đó:
  - `build_feature_vector_from_event(evt)` → vector số (cho ML) + feature dict (để lưu/giải thích).
  - `predict_anomaly(...)`:
    - AE: tính reconstruction error → so với threshold (ví dụ `AE_THRESHOLD`).
    - ISO (nếu có): dùng `decision_function` → so với threshold.

        - Verdict tổng thường là OR của 2 model.
- Gói payload gửi backend:
    - metadata flow (src/dst ip/port, proto, sni, ja3…)
    - score/flag ( `ae_error` , `iso_score` , `is_anomaly` )
    - feature dict được "đẩy lên top-level" để DB query dễ + kèm `features_json` để debug.

**Kênh/"cổng" API mà service gọi**

- HTTP: `POST /api/events` (ingest event từ service → backend).
- Security:
    - Có thể bật HMAC header để chống giả mạo + chống replay:
        - `X-Timestamp` , `X-Nonce` , `X-Signature`
    - Signature = HMAC-SHA256(secret, `ts + "." + nonce + "." + body_bytes_canonical` ) (backend verify đúng format này). filecite☆turn7file5☆L44-L53
    - Backend lưu nonce vào bảng `request_nonces` và chặn reuse (replay). filecite☆turn7file6☆L67-L78 filecite☆turn7file5☆L1-L12

**Luồng dữ liệu (Module 2)**

```
eve.json (TLS events)
  → parse + features
    → ML inference
      → (nếu anomaly) POST /api/events + HMAC headers
```

---

# Module 3

## Backend (FastAPI) – "cổng API" + logic tạo alert/auto-block

Backend là "trung tâm điều phối": nhận event từ python-real-time-service, lưu DB, tạo alert, và (tuỳ chọn) tạo record yêu cầu firewall block.

### 3.1 Các "cổng" (HTTP endpoints)

- Healthcheck: `GET /health` trả `{status:"ok"}` . filecite☆turn7file4☆L26-L29

- Ingest event: `POST /api/events` (service → backend).
  filecite☆turn7file4☆L31-L35🔃
- UI đọc dữ liệu:
  - `GET /api/events?only_anomaly=<bool>&limit=<n>`
    filecite☆turn7file8☆L21-L30🔃
  - `GET /api/alerts?status=<...>&severity=<...>&limit=<n>`
    filecite☆turn7file8☆L33-L45🔃
  - `GET /api/firewall-actions?status=<...>&limit=<n>`
    filecite☆turn7file8☆L48-L57🔃

> Lưu ý: Code có sẵn schema token session + user RBAC (viewer/admin) nhưng ở bản "Read-only UI" hiện tại không có endpoint login public; phần này để mở rộng.
> filecite☆turn7file13☆L46-L55🔃 filecite☆turn7file14☆L29-L53🔃

## 3.2 Ingest + Hardening (HMAC + Anti-downgrade)

- Backend canonicalize JSON body để verify chính xác bytes (tránh sai do Pydantic convert datetime). filecite☆turn7file4☆L34-L45🔃
- Nếu `REQUIRE_INGEST_HMAC=true` → bắt buộc header HMAC; nếu thiếu/sai → reject.
  filecite☆turn7file4☆L46-L58🔃
- Nếu `REQUIRE_INGEST_HMAC=false` nhưng client vẫn gửi `X-Signature` → backend vẫn verify và reject nếu sai (chống downgrade/giả mạo). filecite☆turn7file4☆L59-L72🔃
- Replay protection bằng nonce (unique per scope): reuse nonce → `401 Replay detected`.
  filecite☆turn7file5☆L41-L53🔃

## 3.3 Mapping payload → DB

- Backend nhận TLSEventIn có các nhóm field:
  - Network flow (src/dst ip/port, proto)
  - TLS/JA3 metadata (tls_version, ja3_hash, ja3_string, ja3s_string, sni, …)
  - Feature summary (num_ciphers, weak_cipher_ratio, supports_pfs, groups…)
  - ML outputs (ae_error/ae_anom, iso_score/iso_anom, is_anomaly, verdict)
  - `features_json` (full dict) filecite☆turn7file10☆L49-L97🔃
- DB models tương ứng: `tls_events`, `alerts`, `firewall_actions`, `request_nonces`, `audit_logs`, `users`. filecite☆turn7file7☆L6-L59🔃
  filecite☆turn7file6☆L4-L52🔃

### 3.4 Logic cảnh báo (Alert) + mức độ (Severity)

- Tính `is_anomaly`: ưu tiên `payload.is_anomaly`, nếu không thì OR của `ae_anom` và `iso_anom`. filecite☆turn7file3☆L1-L9
- `compute_severity(...)`:
  - Nếu không anomaly và không có rule flags → không tạo alert.
  - `CRITICAL`: deprecated TLS version hoặc **không có PFS**.
  - `HIGH`: weak cipher hoặc CBC-only.
  - `MEDIUM`: anomaly theo ML.
  - `LOW`: các case rule nhẹ hơn. filecite☆turn7file3☆L16-L28

### 3.5 Auto-block (tạo request cho firewall-controller)

- Nếu `AUTO_BLOCK_ENABLED=true` và severity ∈ {`HIGH`, `CRITICAL`} → backend tạo record `firewall_actions` status = `PENDING`. filecite☆turn7file8☆L1-L16
- Record có integrity fields `hmac_ts`, `hmac_nonce`, `hmac_sig` để firewall-controller verify (backend ký bằng `FW_ACTION_HMAC_SECRET`). filecite☆turn7file3☆L31-L41 filecite☆turn7file6☆L44-L48

#### Luồng dữ liệu (Module 3)

```
POST /api/events (từ service)
  → verify HMAC + nonce
    → insert tls_events
      → compute severity → insert alerts
        → (auto-block) insert firewall_actions(PENDING + signature)
```

---

# Module 4

## Database (MySQL) – nơi “điểm tập trung” dữ liệu

#### Vai trò

- Lưu vĩnh viễn:
  - `tls_events`: tất cả event TLS (đặc biệt anomaly).

- `alerts`: cảnh báo do backend sinh (theo severity).
- `firewall_actions`: hàng đợi lệnh block/unblock để firewall-controller thực thi.
- `request_nonces`: chống replay cho HMAC requests.
- `audit_logs`: (mở rộng) trace hành động admin.
- `users`: (mở rộng) tài khoản + role.

**Luồng "ghi/đọc" DB**

- python-real-time-service **không** ghi DB trực tiếp → chỉ gọi API backend.
- backend:
  - ghi `tls_events`
  - nếu cần → ghi `alerts`
  - nếu auto-block → ghi `firewall_actions`
  - nếu HMAC → ghi `request_nonces`
- firewall-controller:
  - đọc `firewall_actions` (PENDING)
  - update status `EXECUTED/FAILED`

---

# Module 5

## Firewall-controller (worker thực thi block trên firewall bằng iptables)

**Vai trò**

- Là "executor" cho bảng `firewall_actions`.
- Poll DB theo vòng lặp:
  i. SELECT `firewall_actions` WHERE `status='PENDING'`
  ii. Verify integrity (HMAC + max age + expires_at)
  iii. Thực thi iptables (BLOCK/UNBLOCK)
  iv. Update status `EXECUTED` hoặc `FAILED` + error_message

**Cổng/đường dữ liệu**

- Không có HTTP API; nó "ăn DB" (MySQL) như một queue.

- Tác động cuối cùng là *data-plane* firewall: thêm rule DROP vào iptables chain.

**Integrity & chống giả mạo lệnh**

- Backend ký record bằng `FW_ACTION_HMAC_SECRET` → firewall-controller verify trước khi thi hành (nếu không cho phép unsigned). filecite☆turn7file3☆L31-L41
- Nếu action expired (`expires_at` <= now) trước khi thực thi → mark FAILED.

**iptables thực thi**

- Mặc định block bằng:
  - `iptables -I <CHAIN> 1 -s <src_ip> -j DROP`
- UNBLOCK:
  - `iptables -D <CHAIN> -s <src_ip> -j DROP`
- Chain mặc định nên là `INPUT` hoặc `FORWARD` tuỳ firewall đứng ở đâu (gateway/host). (Trong code mặc định là `INPUT`.)

**Reconcilation (tự kiểm tra rule "lạ")**

- Worker định kỳ so sánh "IP đang DROP trong iptables" với "IP bị block theo DB".
- Nếu phát hiện rule DROP "lạ" (thêm tay ngoài hệ thống) thì log cảnh báo; nếu bật option remove unknown thì gỡ rule đó.

---

# Module 6

## Frontend + Nginx gateway (Dashboard)

**Vai trò**

- Dashboard chỉ cần "read":
  - list events/alerts/firewall-actions (qua các API GET ở Module 3).
- Nginx làm reverse proxy:
  - `/api/` → proxy vào backend nội bộ.
  - `location = /api/events` thường nên chặn từ Internet (vì đây là ingest endpoint), chỉ cho python-real-time-service gọi trực tiếp trong mạng nội bộ.

**Cổng (port) người dùng truy cập**

- User truy cập dashboard bằng HTTP port (ví dụ) `8080` trên máy sensor:
  - `/` → UI
  - `/api/events|/api/alerts|/api/firewall-actions` → proxy backend (đọc dữ liệu)

---

# Module 7

## Tổng luồng vận hành thật sự (cổng nào, dữ liệu đi như nào)

### 7.1 Data-plane (lưu lượng thật)

1. Client ↔ Server TLS đi qua switch/firewall như bình thường.
2. Switch mirror vào SPAN → NIC monitoring sensor nhận toàn bộ frame (promisc).
3. Suricata sniff traffic → log `eve.json` (TLS events).

### 7.2 Control-plane (hệ thống IDS)

1. python-real-time-service tail `eve.json`
2. Feature extraction + ML → nếu anomaly:
3. Gửi HTTP `POST /api/events` vào backend (cổng ingest) + optional HMAC headers.
4. Backend:
   - verify HMAC/nonce
   - insert `tls_events`
   - compute severity → tạo `alerts`
   - nếu auto-block → insert `firewall_actions(PENDING + signature)`
5. Firewall-controller poll DB:
   - verify chữ ký action
   - thực thi iptables DROP
   - update `firewall_actions` → `EXECUTED` / `FAILED`
6. Frontend (dashboard):
   - Người dùng mở web → gọi `GET /api/events`, `GET /api/alerts`, `GET /api/firewall-actions` để xem.

**Pipeline tóm tắt**

```
[SPAN traffic] → Suricata → eve.json
  → python-real-time-service → (HTTP POST /api/events) → Backend → MySQL
    → (auto-block) MySQL.firewall_actions → firewall-controller → iptables DROP
  → Dashboard (HTTP GET /api/*) đọc lại từ Backend
```

---

# Module 8

## Notes kỹ thuật / vận hành (thực tế triển khai)

- **Tách máy sensor vs máy firewall**:
  - Sensor: bắt traffic + chạy Suricata + pipeline ML + backend + DB + dashboard (lab).
  - Firewall host: chạy firewall-controller (privileged + host network) để thao tác iptables.
- **Hardening khuyến nghị**:
  - Không expose trực tiếp backend port 8000 ra Internet (chỉ đi qua Nginx).
  - Bật `REQUIRE_INGEST_HMAC=true` để ingest chỉ nhận từ service hợp lệ. filecite☆turn7file1☆L15-L18 filecite☆turn7file4☆L46-L58
  - Set `FW_ACTION_HMAC_SECRET` và để `ALLOW_UNSIGNED_FW_ACTIONS=false` để tránh ai đó tự insert DB row mà firewall thực thi. filecite☆turn7file1☆L24-L26 filecite☆turn7file3☆L31-L41
- **Tạo severity/alert dựa trên rule flags + anomaly**: rule flags làm “explainable” (vì sao HIGH/CRITICAL), ML làm “catch unknown patterns”. filecite☆turn7file3☆L16-L28