Implementation Report

Zero Trust Architecture (ZTA) and Machine Learning for UAV Security

Name: Yadunandan N
Reg. No: 251091010013
Program: M.Tech in Cybersecurity, MIT

1. System Overview

This implementation operationalizes a Zero Trust Architecture (ZTA) combined with ML-driven detection for UAV fleets operating in an IoBT/edge environment. The design enforces continuous verification ("never trust, always verify"), least-privilege access, and behavior-based anomaly detection.

- **Scope:** UAVs, Ground Control Station (GCS), Command & Control (C2) links, edge nodes, secure data lake.
- **Goals:** Strong identity, adaptive access control, near–real-time anomaly detection, auditability, and resilience.
- **Assumptions:** Each UAV has a TPM/secure element; links support mutual auth; time is synchronized (GNSS/secure NTP).

2. Architecture and Components

Core ZTA roles

- **Policy Engine (PE):** Calculates dynamic trust score; decides *allow/deny/step-up*.
- **Policy Administrator (PA):** Issues short-lived tokens/policy enforcements.
- **Policy Enforcement Point (PEP):** Enforces decisions on data plane (sidecars/ingress gateways on GCS/edge/UAV).

Identity & Credentials

- X.509/mTLS between UAV $\leftrightarrow$ GCS; SPIFFE IDs for workload identity.

- Hardware root-of-trust (TPM) for key storage & attestation.

Telemetry Bus

- Signed logs (syslog/OTLP) from UAV/edge/PEP into a message broker (e.g., NATS/Kafka).
- Features stream to the ML service for scoring.

ML Detection Service

- Online model server (RF/XGBoost/One-Class SVM) with sliding-window features.
- Outputs *risk score* $\in [0, 1]$ to the PE.

3. Data & Feature Pipeline

Collection (UAV/PEP/Edge):

- Network: RTT, jitter, packet loss, retransmissions, unusual ports.
- Control: sudden waypoint changes, altitude/speed deviations, command frequency spikes.
- Integrity: attestation state, firmware hash, config drift.

Pre-processing (Edge):

1. Deduplicate & validate signatures.
2. Normalize/standardize numeric fields.
3. Build rolling features (e.g., last 60s mean/std/min/max).
4. Encrypt in transit (mTLS) and at rest (AES-GCM).

Feature Examples (per 30–60s window):

- μ, σ of uplink/downlink bitrate; ratio of control vs. payload frames.
- Command entropy; unique command types; burstiness index.
- GNSS vs. IMU drift; expected vs. actual flight vector residual.

4. ML Models (Training & Online Scoring)

Training (offline):

1. Label historical windows: *benign* vs. *attack* (spoofing, jamming, replay, unauthorized commands).
2. Train baseline models:
   - **Random Forest/XGBoost**: tabular supervised detection.
   - **One-Class SVM or Isolation Forest**: anomaly detection for *unknown* patterns.
3. Calibrate thresholds (ROC/PR curves) to minimize mission-impacting false positives.
4. Export artifacts (model + scaler + feature schema) with versioning.

Online scoring (edge or GCS):

1. Compute feature vector per window, validate schema.
2. Score with current model version.
3. Emit `risk_score` and `explanation` (e.g., top features via SHAP).

5. Dynamic Trust & Policy Evaluation

Trust score $T \in [0, 100]$:

$$T = w_1 \cdot \text{Identity} + w_2 \cdot \text{Posture} + w_3 \cdot (1 - \text{risk\_score}) + w_4 \cdot \text{Context}$$

- **Identity:** mTLS + attestation freshness.
- **Posture:** firmware integrity/config compliance.
- **Context:** geo-fence, mission phase, time, operator role.

Decision policy (example):

- $T \geq 80$: allow + continuous monitoring.
- $60 \leq T < 80$: step-up (human-in-the-loop confirm / extra auth).
- $T < 60$: deny; trigger containment (rate-limit, command quarantine, RTH).

6. Access Control & Segmentation

Principles: least-privilege, micro-segmentation, deny-by-default.

- Sidecar PEPs enforce Layer-7 policies (only allowed API/verbs).
- Short-lived tokens (JWT/OAuth MTLS-bound) issued by PA.
- Per-mission RBAC/ABAC (attributes: role, UAV type, mission ID, geo-zone).

7. Deployment Steps (Practical Checklist)

1. **Provision identities** for UAV/GCS/edge (PKI, SPIFFE IDs); enable TPM-based key storage.
2. **Install PEPs** (ingress gateway/sidecar) on UAV OS image, GCS services, and edge nodes.
3. **Stand up PE/PA** with initial deny-all baseline; define policy schemas and trust function.
4. **Telemetry bus** (NATS/Kafka) + collector agents; verify signed logs and mTLS.
5. **Feature/Model service** at edge or GCS; deploy v1 model and health checks.
6. **Progressive rollout:** canary subset of UAVs → blue/green for full fleet.
7. **Runbooks:** incident response (containment, RTH), model rollback, key rotation.

8. Resilience & Safety Controls

- **Fail-safe flight:** on deny/timeout, UAV executes RTH/hover protocol.
- **Jamming/spoofing detection:** spectrum anomalies → increase risk, switch to hardened link.
- **Key rotation:** frequent, automated; revocation via CRL/OCSP stapling.
- **Audit:** append-only log store; periodic attestation sweeps.

9. Evaluation Metrics

- Detection: Precision/Recall/F1, AUROC, mean time to detect (MTTD).
- Operations: latency overhead (PEP/PE), command round-trip times.
- Security: policy coverage, blocked unauthorized attempts, key-compromise MTTR.
- Energy/compute: CPU/GPU load on UAV, watt-hours per hour of scoring.

10. Configuration Snippets (Illustrative)

Example ABAC policy (YAML-like pseudo):

```
policy:
  name: uav-command-execute
  subject: spiffe://fleet/uav/${UAV_ID}
  action: POST:/v1/commands/*
  resource: c2-api
  conditions:
    - attestation.fresh == true
    - firmware.hash in approved_hashes
    - geofence.contains(telemetry.position)
    - trust_score >= 80
  effect: allow
```

PEP rate-limit on step-up:

```
when trust_score in [60,79]:
  allow actions=[telemetry.get]
  deny  actions=[command.execute]
  require step_up = operator_confirm
```

12. Notes for Integration

- Keep models lightweight for edge; consider quantization or distilled models.
- Version & sign all models and policies; PE must reject unsigned artifacts.
- Treat explanations (top features) as operator cues, not automatic override.