STRIDEtastic(2): Threat Modeling Framework

Francisco Wulf
Pontificia Universidad Católica de Chile
francisco.wulf@uc.cl

***Abstract*—This document presents the second research increment after the initial introduction (*Increment 1*) in the investigation *STRIDEtastic: Threat Modeling and Security Evaluation of Meshtastic Networks through Practical and Theoretical Attacks*. This increment focuses on establishing the theoretical framework for the study, defining STRIDE and outlining the methodology to apply STRIDE to Meshtastic within the context of Cyber-Physical Systems (CPS). We provide a detailed mapping of system components, data flows, STRIDE threat enumeration, prioritized mitigations and a laboratory validation plan for controlled experiments. The STRIDE methodology and procedural steps used here are sourced from the STRIDE-for-CPS paper; Meshtastic-specific facts and the Increment-1 analysis are taken from the provided Meshtastic report (Increment 1). [1], [2]**

I. Introduction

Increment 2 establishes the theoretical and technical foundation for the research regarding threat modeling. In particular, this increment adopts and applies the STRIDE-for-CPS methodology as formalized by Khan et al. (2017) and adapts it to the Meshtastic system characteristics documented in Increment 1 [1], [2]. The goal is to produce a repeatable methodology and a prioritized set of mitigations tailored to Meshtastic networks (low-power LoRa mesh plus management interfaces).

II. System Description

A. Scope and Objectives

The threat model covers devices running Meshtastic firmware (LoRa-based mesh nodes), their local interfaces (BLE, serial, TCP/HTTP), gateway/infrastructure components (MQTT broker, web/python clients) and external entities (GPS satellites, human users). Primary security objectives include confidentiality, integrity, availability, authentication/authorization and non-repudiation for messages and node configuration.

B. DFD Components (IDs)

The components identified from the provided DFDs are listed below in Table I.

TABLE I: DFD components and classifications

ID	Type	Description
EE1	EE	GPS_SAT (GNSS signal source)
EE2	EE	USER (human agent: Android/iOS/Web/Python user)
P1	P	NODE_A (user's device)
P2	P	NODE_B (0-hop via LoRa)
P3	P	NODE_C (multi-hop via LoRa)
P4	P	NODE_D (0-hop via MQTT broker)
P5	P	NODE_E (multi-hop via MQTT broker)
P6	P	BROKER_MQTT (MQTT broker / cloud or private broker)
P7	P	PYTHON_API_SDK (integration client)
P8	P	ROUTING_APP (routing logic in node)
P9	P	TEXT_MESSAGE_APP (message handling)
P10	P	POSITION_APP (position management)
P11	P	TELEMETRY_APP (telemetry)
P12	P	PRIVATE_APP (for proprietary/non-official applications)
P13	P	NODEINFO_APP (node metadata service)
P14	P	ADMIN_APP (administrative functions)
DS1	DS	NODEDB (local peer database / pubkeys / IDs)
DS2	DS	MESSAGE_CACHE (outgoing/incoming message queue)
DS3	DS	CONFIGURATION (device configuration)
UI1	UI	ANDROID_UI
UI2	UI	APPLE_UI
UI3	UI	WEB_CLIENT
UI4	UI	BASE_UI
UI5	UI	MUI (Meshtastic UI for standalone devices)
DF1	DF	LoRa (radio link: peer-to-peer mesh)
DF2	DF	BLE (local management link)
DF3	DF	SERIAL (USB / UART local interface)
DF4	DF	TCP (IP connectivity)
DF5	DF	MQTT (pub/sub over TCP/TLS)
DF6	DF	GPS (GNSS data flow)
DF7	DF	Broker links (MQTT broker ↔ clients)

C. Trust Boundaries

Following STRIDE-for-CPS guidance, we define trust boundaries to delimit what is considered inside the trusted domain versus external/untrusted:

- **Node-local boundary:** hardware + firmware + local storage on each NODE_X. Access via serial/BLE implies crossing into a higher trust level.
- **Mesh boundary:** LoRa peer-to-peer communications traverse this boundary; other nodes are partially trusted and treated as potential attackers if compromised.
- **Infrastructure boundary:** Broker/MQTT and cloud/-gateway services — considered outside the mesh trust by default unless they are privately operated and controlled.
- **User device boundary:** mobile clients and python SDKs are user-controlled and may be compromised.
- **External systems:** GPS satellites are considered untrusted with respect to authenticity (GNSS spoofing/jamming concerns). [1], [2]

III. Threat Modeling Framework

A. STRIDE Framework

STRIDE is used here as the primary lens for cyber-threat enumeration (Spoofing, Tampering, Repudiation, Information disclosure, Denial of Service, Elevation of privilege).

The STRIDE-for-CPS paper provides (i) the motivation for choosing STRIDE as a relatively lightweight and component-focused approach, and (ii) a five-step methodology suitable for CPS threat modeling: decompose system, plot DFDs, analyze threats, identify vulnerabilities, plan mitigations [1].

B. STRIDE in Cyber-Physical Systems

Khan et al. specifically adapt STRIDE to CPS by emphasizing:

- physical-layer threats (jamming, antenna tampering, GNSS spoofing),
- timing/real-time attack vectors (replay/time-shifting that impact physical control loops),
- resource constraints on embedded devices (cryptographic cost, logging limits, secure boot, secure key-storage),
- the need to perform both *STRIDE-per-element* and *STRIDE-per-interaction* analyses because certain threats only manifest in interactions. [1]

C. Per-element vs Per-interaction approach

As shown in Khan et al., STRIDE-per-element catalogs the threats that are possible on each DFD element type (entity/process/dataflow/datastore) while STRIDE-per-interaction analyzes each origin-destination tuple to capture threats that cross components. Both approaches are complementary; for Meshtastic we perform a per-element sweep and then a targeted per-interaction enumeration for critical flows (LoRa mesh, MQTT gateway, BLE/TCP interfaces). [1]

IV. Methodology for Applying STRIDE to Meshtastic

A. Threat Modeling Steps

We adopt the five-step procedural flow from Khan et al., instantiated for Meshtastic:

1) **System decomposition**: list components and interfaces (Table I). [2]
2) **Plot DFDs**: create DFDs for (a) overview and (b) peer-to-peer communications (these are the diagrams you supplied).
3) **Analyze threats**: perform STRIDE-per-element and STRIDE-per-interaction as in the reference methodology. [1]
4) **Identify vulnerabilities**: map missing security properties (authN / authZ / confidentiality / integrity / availability / non-repudiation) back to each component, guided by the Meshtastic findings in Increment 1. [2]
5) **Mitigations and plan**: propose mitigations and design lab validation experiments.

V. Data Flows and Threat Mapping

In Figures 1 and 2 we map the main data flows (DFx) to source/destination. In Table II we also include payload type and primary STRIDE concerns. This is a direct instantiation off the STRIDE-per-interaction idea used in the STRIDE-for-CPS paper applied to the Meshtastic DFD.

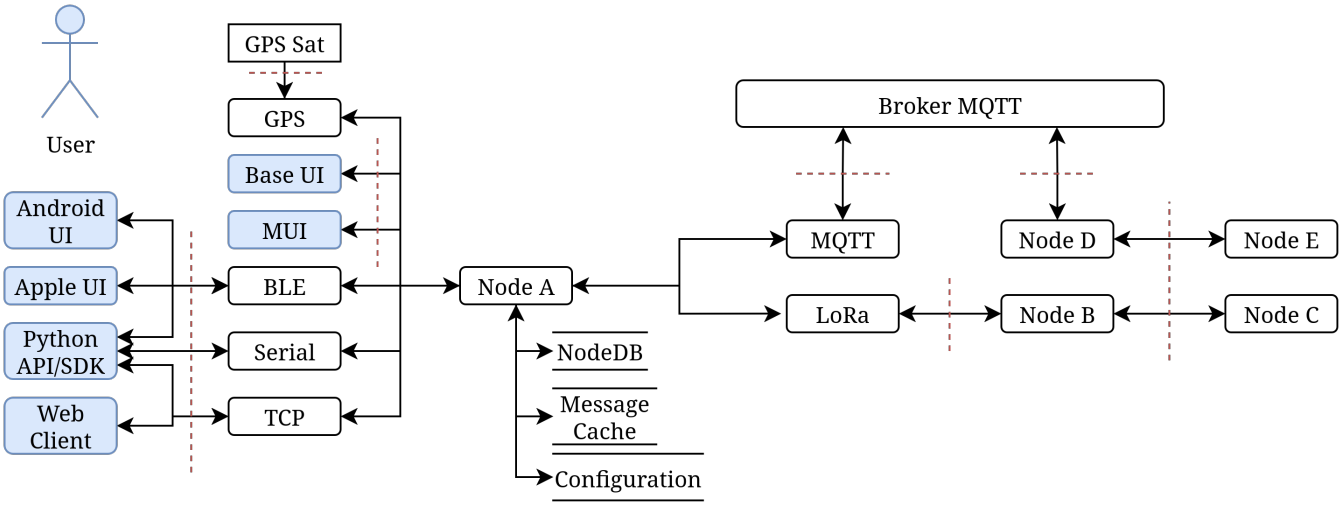

Fig. 1: Data Flow Diagram for a generic Meshtastic system.

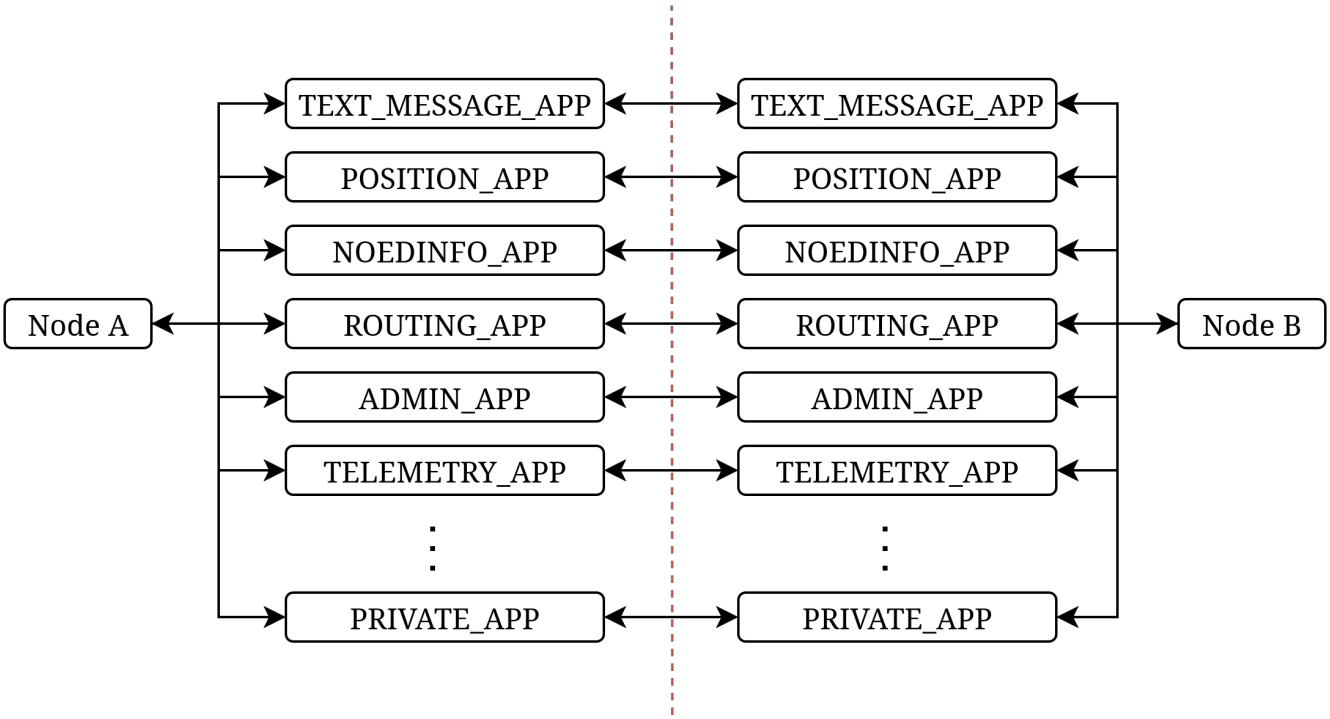

Fig. 2: Data Flow Diagram of Two Connected Nodes.

DF	From → To	Payload	Primary STRIDE concerns
DF1	Node A ↔ Node B (LoRa)	Mesh messages, routing control, telemetry	Spoofing (fake node), Information Disclosure (eavesdrop; headers in clear), Denial (jamming), Tampering (message modification / replay)
DF2	Node ↔ Mobile (BLE)	UI config, credentials, messages	Spoofing (fake peripheral), Tampering (MitM during insecure pairing), Information Disclosure (insecure pairing modes)
DF3	Node ↔ Host (Serial/USB)	CLI commands, firmware updates	Elevation of privilege (console access), Tampering (flash firmware), Repudiation (local commands without audit)
DF4	Node/Gateway ↔ Internet (TCP/HTTP)	Bridge messages, telemetry	Information Disclosure (if HTTP/plain), Spoofing, Tampering, DoS
DF5	MQTT Broker ↔ Clients	Topics: messages, status	Spoofing (publish as other), Information Disclosure (if broker receives decrypted payloads, or keys are known), Tampering, DoS (topic floods)
DF6	GPS → Node	Position / time	Spoofing (GNSS), Denial (jamming), Integrity of time stamps (affects position-based features)
DF7	Broker ↔ Web/Python	UI messages, API calls	Authentication (API keys, TLS), Information Disclosure, Tampering, Repudiation (insufficient signed logs)

TABLE II: Data flows, payloads and STRIDE focal points.

VI. STRIDE Enumeration: Component-level

A. Spoofing

We classify spoofing attacks both by **authentication vector** (what cryptographic or keying primitive the flow relies on) and by **complexity / sophistication** (simple vs advanced / managed attacks).

By authentication mechanism:

a) Channel PSK (symmetric channel keys):

- If the channel PSK is known (default PSK or leaked), **all spoofing attacks** that rely on forging frames for that channel become trivial (see lists below).

b) PKC (DM / Admin signatures):

- **Stolen key pair — DM spoofing**: theft of a node's private key (used for DMs) enables an adversary to craft DMs that verify correctly and appear to originate from the victim.
- **Stolen key pair — Admin Module spoofing**: compromised private keys used by admin/control messages permit full impersonation of administrative operations.

By complexity:

c) Simple:

- **One-shot noisy attacks**
  - Spoof Node Info
  - Spoof Position
  - Spoof Broadcast Text Message
  - Spoof DM via downgrade attack
  - Spoof Traceroute
  - Spoof other modules (any app running on port ≥ 256)
- **Simply-managed noise attacks**
  - Reduce hop count and choose a favorable interface to inject spoofed frames (lower hop count reduces spread and detection surface).
  - Use basic noise shaping to limit collateral reception.

d) Advanced:

- **Node-info & position smart spoofing**
  - Immediately inject spoofed packets when legitimate packets are observed (on-the-fly).
  - Periodic spoofing to simulate continuity of false identity/state.
- **Spoof DM on-the-fly over re-transmitting interfaces**
  - When a DM packet is observed, delay its relay and inject a spoofed DM (downgrade + timing manipulation). This leverages UI behaviors that mark chats as "secure" based only on the latest message.
- **Network discovery via traceroute spoofing**
  - Manipulate traceroute responses to misrepresent topology and expose routes with higher concentration of victims.
- **Managed noise attacks (formalized)**
  - Treat interfaces as abstract channels (LoRa, MQTT, TCP, Serial). Define a malicious interface as one controlled by the adversary; define adjacent interfaces as those directly connected.
  - Let N be the set of all nodes. Partition N into $N_1 \cup N_2 \equiv N$ such that nodes in N_1 received spoofed packets in P while nodes in N_2 do not.
  - Define $N_A \subseteq N$ as the group(s) of nodes targeted to receive identical malicious packet(s). The attacker seeks an interface and physical placement to maximize $|N_A \cap N_1|$ and minimize exposure $|\bar{N_A} \cap N_1|$.
  - Techniques include:
    - reduce hop_limit to restrict propagation,
    - send via carefully chosen physical interfaces to shape reception footprint,
    - place malicious LoRa interface physically adjacent to target nodes,
    - exploit sparse regional deployments to achieve high signal-to-noise for target nodes while avoiding others,
    - use compromised adjacent interfaces (e.g., MQTT gateway) to inject packets into limited mesh partitions.
  - Adversaries may range from hobbyists (simple antennas) to well-resourced actors with directional, high-precision RF equipment.

B. Tampering

Tampering describes unauthorized modification of packets, configuration and stored data. In Meshtastic, tampering feasibility depends on key material and whether packets travel encrypted or decoded.

- **Conditions enabling tampering:**
  - The attacker knows the channel PSK.
  - The packet is relayed in decoded form by an intermediate that has the key.
- **Advanced tampering (rule-based):**
  - Rule-based tampering engines select targets by node identity, receiving and transmitting interfaces, port numbers and packet types; they can modify payload fields or selectively drop/alter packets to influence application behavior.
- **Cache-clearing tampering:**
  - If an adversary knows the user is not connected to the node via a companion UI (or is able to jam the connection), it can flood the node with low-value messages to force eviction of important pending messages from MESSAGE_CACHE (denial/availability effect combined with tampering).

C. Repudiation

Practical repudiation vectors related to the attacks above include:

- **Lack of signed logs**: nodes and brokers do not produce signed audit trails, attackers can deny actions (e.g., claim messages were spoofed when attacker performed privileged operations).
- **UI downgrade/ambiguity**: UIs that display only the latest message or lack cryptographic provenance enable

attackers to create messages that appear to be user-originated and be later denied by a legitimate user.
- **Log manipulation by compromised node**: a node with admin privileges or filesystem access can reboot the device, effectively removing logs and thus, evidence.

D. Information Disclosure

The following assets and vectors are of particular concern for information disclosure in Meshtastic.

- **Node info and public keys**: public metadata and pubkeys can be harvested by passive sniffing.
- **Position and tracking**: periodic position broadcasts and header metadata allow movement profiling and tracking.
- **Sensor telemetry**: temperature, battery, and other sensor metrics may leak operational context.
- **Signal metrics**: RSSI and SNR leakage enable signal-based tracking and physical-layer correlation attacks. This effectively enables mobile node tracking.
- **Key stealing**: extraction of key material from NODEDB or backups results in full loss of confidentiality and impersonation capabilities.
- **Network Topology Intelligence**: by monitoring the network, adversaries can gain intelligence in terms of mesh topology, as packets transmit routing information, like sender node and gateway nodes. Also, traceroute payloads give away information about network topology to the whole channel.

E. Denial of Service

DoS vectors include classical and mesh-specific attacks:

- Packet flooding (broadcast storms).
- Hop exhaustion / sinkhole attacks: inject tampered header packets that tell no hops remain, disabling retransmission.
- Next-hop poisoning: forge routing metadata to redirect traffic or create loops.
- Channel jamming: RF noise to block LoRa channels.
- Malformed packet injection: exploit parser weaknesses or cause excessive processing.
- Wormhole attacks: capture and replay packets across distant segments to confuse topology and cause constant flooding.
- Amplified wormhole attack: manipulate hop_limit and message_id fields (e.g., set hop_limit high and alter message_id to trigger retransmission loops) to induce excessive retransmissions and amplification.

F. Elevation of Privilege

Privilege escalation vectors include:

- Legacy admin channel exploitation: they use PSK, so compromising one admin-capable node is all an adversary need to gain full admin control.
- Local console / serial access without authentication: allows firmware flash and configuration changes like admin key injection.
- Compromised LAN attack: TCP does not authenticate, so any adversary who can position itself inside the LAN, can effectively have admin access to the node.
- Compromised MQTT broker: Attackers can sniff and inject via the broker. Worse, Meshtastic default configuration allows relaying packets from unknown sources, meaning that one could gain access to a private MQTT broker by doing packet sniffing and injection adjacent to at least one targeted MQTT gateway node.

STRIDE: ASSET — VECTOR — STRIDE CATEGORY

Asset	**Attack vector (brief)**	**STRIDE category**
Mesh headers / metadata	Passive sniffing with SDR / LoRa receiver (headers in clear)	Information Disclosure
Channel payload (PSK)	Use default PSK / stolen PSK to inject frames	Spoofing / Tampering / Information Disclosure
DM private keys	Theft of private key from NODEDB or backup	Spoofing (DM impersonation) / Info Disclosure
Admin private keys	Theft of admin keypair	Spoofing (admin impersonation) / Elevation of Privilege
POSITION_APP data	Fake GNSS / injected position frames	Spoofing / Information Disclosure
Traceroute responses	Traceroute sniffing to reveal topology	Information Disclosure / Tampering
MESSAGE_CACHE	Flood to overflow / evict important messages	Denial of Service / Tampering
Node firmware	Flashing malicious images via serial	Tampering / Elevation of Privilege
MQTT broker payloads	Flooding / Broker configured with decrypted payloads / broker compromise	Denial of Service / Information Disclosure / Tampering / Spoofing
LoRa physical channel	Directional jamming / wormhole / packet injection / packet on-the-fly manipulation / managed noise	Denial of Service / Tampering / Spoofing

TABLE III: Selected asset-vector-stride mappings (illustrative; not exhaustive).

VII. Risk Assessment and Prioritization

We adopt a qualitative Likelihood $\times$ Impact approach informed by the STRIDE-for-CPS guidance (which stresses that spoofing and tampering often have severe downstream physical consequences) and the Meshtastic technical profile.

VIII. Mitigations and Security Controls

Mitigations below are mapped to the principal STRIDE categories and tailored to Meshtastic constraints (low-power radios, need to relay undecipherable headers, support for both symmetric channels and PKC DMs).

A. Spoofing mitigations

- Sign messages with Ed25519 for identity binding.
- Broker and client mutual authentication (TLS client certs / strong API keys).
- BLE pairing: enforce random one-time PIN and explicit user confirmation flows.

B. Tampering mitigations

- Signed firmware images + secure boot (use and develop platforms that supports it).
- Message integrity: Ed25519 signature.
- Config integrity: sign configuration blobs and enforce verification on load.

C. Repudiation mitigations

- Signed events/messages; mesh developers should remote archival of signed logs when possible (i.e. meshtasticd).
- Forensic-friendly logging formats and retention policies in broker and node management backends.
- Use tools to collect network intelligence (sniffer, dissector, etc.).

D. Information disclosure mitigations

- Default to encrypted payloads on MQTT integrations; avoid broker-side decryption unless strictly required and access-controlled.
- Secure storage for keys (encrypted at rest; hardware-backed keystore recommended when possible).
- Verifiable identities for addressing instead of hardware-derived MAC addresses.
- Use tools to collect network intelligence and mapping information disclosure.

E. Denial of Service mitigations

- Rate limiting and topic quotas at the broker; policing of gateway-originated messages to avoid vicious cycles that cause zero-hopping or ban of legitimate nodes (meshtastic MQTT broker boilerplate).
- Node-side message cache limits and eviction policies.
- Jamming detection heuristics (monitor RSSI/noise floor, repeated packet loss patterns) and failover behaviors (graceful degradation). [1]
- Use tools to collect monitor network status.

F. Elevation of Privilege mitigations

- Protect administrative operations with local credentials and session timeouts; avoid exposing privileged operations over unauthenticated TCP/HTTP by default.
- Adopt principle of least privilege for APIs and CLI commands exposed over networks.

IX. Laboratory Validation Plan (Controlled Experiments)

The STRIDE-for-CPS paper demonstrates the importance of lab validation to test threats and mitigations; we adapt the same philosophy to Meshtastic testbeds.

A. Testbed components

- 5 Meshtastic nodes with firmware v2.7.4.
- Cloud MQTT Broker.
- Mobile clients (Android/iOS), Web client.
- STRIDEtastic application enabling attack development.
- Orange Pi Zero 2W with Waveshare LoRa module for sniffing, spoofing, tampering, etc.
- Meshtastic node running patched firmware (or LoRa interface) connected to Orange Pi for sniffing and packet injection.

B. Representative controlled attack scenarios

1) **Passive LoRa eavesdrop:** Use MQTT client and LoRa interfaces to eavesdrop packets from both MQTT broker and LoRa spectrum. Decode and decrypt packets with known keys.
2) **Wireshark and Meshtastic:** Enable pcap download and dissection in Tshark/Wireshark.
3) **Node identity, position and text message spoofing:** Use interfaces to inject specially crafted broadcast packets, enabling packet spoofing.
4) **Advanced spoofing techniques:** Reactive, periodic, managed noise.
5) **Network geometry discovery:** Map and graph network geolocation and topology.
6) **Tampering:** Integrate tampering workflow, tamper packet hop limit.
7) **Denial of Service:** Develop flooding, hop exhaustion and selective packet dropping attacks.
8) **TCP interface:** Discover nodes in the network and use TCP interface to gain network intelligence.

X. Assumptions and Limitations

- This threat model assumes the correctness of the Meshtastic findings in Increment 1 (packet format, encryption modes, interface defaults). All Meshtastic-specific claims are grounded in that document. [2]
- Some mitigations (secure elements, secure boot) depend on hardware support that is not universally available across all Meshtastic hardware variants.
- Physical-layer threats (jamming, GNSS spoofing) cannot be fully eliminated; the goal is detection and graceful degradation following CPS STRIDE guidance. [1]

- Risk scoring here is qualitative; further metrics collection in the lab will enable quantitative scoring in subsequent increments.

XI. Next Steps

1) Describe STRIDEtastic as a framework for secure Meshtastic applications development.
2) Develop attacks and attack scenarios.

References

[1] R. Khan, K. McLaughlin, D. Laverty and S. Sezer, "STRIDE-based Threat Modeling for Cyber-Physical Systems," *Proc. IEEE*, 2017.

[2] F. Wulf, "STRIDEtastic(1): Meshtastic and Security" (Increment 1 — Meshtastic system description and security analysis), Pontificia Universidad Católica de Chile, 2025.

APPENDIX

Component	Top STRIDE Threats	Suggested Mitigations
NODE (firmware + apps)	Spoofing, Tampering, DoS, Info Disclosure	Firmware signing, secure storage, E2E for private messages, rate limiting, signed logs
NODEDB / CONFIGURATION	Tampering, Info Disclosure	Encrypted at rest, integrity checks, access controls
LoRa (DF1)	Eavesdrop, Spoof, Jamming	E2E encryption, anomaly detection, link quality monitoring, message anti-replay
BLE / Serial (DF2/DF3)	MitM, Elevation	Authenticated pairing, disable privileged commands by default, require local auth for admin tasks
BROKER_MQTT	Spoofing, Tampering, Info Disclosure	TLS, client certs, topic ACLs, limit retained payloads, avoid broker-side decryption for sensitive channels
GPS (EE1)	Spoofing, Jamming	GNSS cross-checks, plausibility filters, multi-source positioning if available

TABLE IV: Component-level summary: top threats and mitigations