Comprehensive Research Roadmap

Memory Attacks and Defenses in Multi-Agent LLM-Controlled UAV Systems under Energy Constraints

Research Plan for Ibrahim Odat

November 27, 2025

Purpose of This Document

This roadmap integrates the advisor-approved proposal, the refined conceptual roadmap, and the execution plan into one coherent document. It is written as if I am the primary PhD researcher with 20+ years of experience planning a focused but novel project, with a real submission deadline. The goal is to keep the scope ambitious enough to be impressive, but constrained enough to finish.

I will maintain this file as the authoritative plan for:

- the **research problem** and its novelty,
- the final **research questions** and **objectives**,
- a phased **implementation and evaluation roadmap**,
- concrete **deliverables** and **metrics**.

1 Problem Statement and Novelty

Modern large language model (LLM) agents rely heavily on long-term memory to operate over extended missions and partially observable environments. In multi-agent settings, such as multi-UAV autonomy, this memory is often *shared* across agents and spans multiple modalities: episodic logs, semantic rules, and procedural skills. While recent work has shown that LLM agents can be vulnerable to memory injection and poisoning attacks in purely digital domains (e.g., web tasks, recommendation), there is little understanding of how these attacks manifest in **physical**, **energy-constrained** multi-UAV systems controlled by LLM-based agents.

This project studies a multi-agent LLM controller for two UAVs (PX4 SITL) that uses:

- **Episodic memory** to store structured logs of missions, detections, and outcomes.
- **Semantic memory** to capture higher-level hazard rules, geofences, and energy heuristics.
- **Shared memory** across a *supervisor* and two *worker* agents, enabling cross-mission and cross-agent reuse of experience.

An adversary is assumed to have only **query-level access** to the system (no direct DB access), but can interact with the agents and influence which memory entries are written and later retrieved—in the spirit of query-only memory attacks (e.g., MINJA-style). The central question is how such attacks impact **safety**, **mission success**, and **energy-constrained endurance**, and how a lightweight defense layer around memory changes this trade-off.

The novelty lies in:

- bringing **structured, multi-layer memory** (episodic/semantic) into a **multi-UAV LLM control stack** with PX4 SITL;
- designing **query-only memory poisoning attacks** against this stack, focusing on safety and endurance under energy constraints;
- implementing and measuring a **defense layer** (integrity tags, semantic validation, cross-agent checks) and quantifying the resulting *multi-agent security tax* (overhead in latency, tokens, and simulated energy);
- distilling **design principles** for memory in energy-constrained multi-agent UAV autonomy stacks.

2 Final Research Questions

The project is structured around three research questions.

RQ1. Impact of memory poisoning.
How do targeted *episodic and semantic memory poisoning attacks*, mounted by a **query-only adversary**, affect **safety**, **mission success**, and **simulated energy usage** in a multi-UAV LLM-based control system?

RQ2. Effectiveness and cost of defenses.
To what extent can a **combined defense layer**—using *integrity tags* on episodic memory and *semantic validation / cross-agent consistency checks* for shared rules—**detect and mitigate** these attacks, and what is the resulting **multi-agent security tax** in terms of additional **latency**, **token usage**, and **energy overhead**?

RQ3. Design principles for memory in multi-UAV LLM autonomy.
Based on these experiments, what **design principles** for structuring and sharing memory in **energy-constrained multi-UAV LLM autonomy stacks** can be derived?

3 Project Objectives

The following objectives are chosen to be achievable under a realistic deadline while supporting the above research questions.

O1. System and memory implementation.
Design and implement a **multi-UAV testbed in PX4 SITL** with a **supervisor** and two

worker LLM agents, connected to **MAVSDK** and **perception/sensor APIs**, and equipped with shared **episodic** and **semantic** memory stores.

O2. **Attack suite (query-only memory poisoning).**
Develop a suite of **realistic memory poisoning attacks** targeting the episodic and semantic layers—e.g., *fake obstacle / low-SOC episodes*, *stale hazard replays*, and *corrupted hazard / energy heuristics*—and integrate them into a controllable attack harness matching a **query-only attacker**.

O3. **Defense layer around memory.**
Design and integrate a **lightweight memory-defense layer** around the system, including **integrity tagging** for episodic entries and **semantic consistency / support checks plus cross-agent verification** for shared rules, with defenses that can be **toggled on/off** for evaluation.

O4. **Empirical evaluation and design insights.**
Conduct a **systematic experimental campaign** in simulation to quantify (i) the **impact of attacks** (RQ1) and (ii) the **effectiveness and cost of defenses** (RQ2), and synthesize **practical design recommendations** for memory in multi-agent LLM-controlled UAV systems (RQ3).

4 Planned Contributions

If successfully executed, this project should support the following contributions in the final paper:

- A **reference architecture** for a multi-agent LLM-controlled, energy-aware multi-UAV system with explicit episodic and semantic memory.
- A concrete **attack taxonomy** and **attack harness** for query-only memory poisoning in this setting.
- A **defense bundle** around memory, including integrity tags and semantic / cross-agent validation, integrated into the decision loop.
- A quantitative study of the **multi-agent security tax**: how much latency, token usage, and simulated energy overhead defenses add relative to their security gains.
- Empirically grounded **design principles** for memory in multi-UAV autonomy systems.

5 System Overview and Threat Model (Working Sketch)

This section summarizes the assumed system and threat model at a high level. Details will be refined as the implementation progresses.

System Overview

- Two UAVs simulated in **PX4 SITL**, controlled via **MAVSDK**.
- A **Supervisor** agent receives high-level mission goals and decomposes them into tasks.
- Two **Worker** agents (Drone 1 and Drone 2) execute tasks via tool calls to MAVSDK and perception/sensor APIs.
- **Perception/Sensor APIs** provide scene-level detections and sensor readings (e.g., SOC, GPS, distance to home).
- A shared **Episodic Memory** store logs time-stamped episodes: state, action, outcome, context.
- A shared **Semantic Memory** store maintains higher-level rules (hazards, no-fly zones, energy constraints).
- An **Energy Model** on top of SITL computes simulated energy usage and SOC over time from distance/time or velocity logs.

Threat Model (Query-Only Attacker)

- The attacker cannot directly edit the memory database or system binaries.
- The attacker can issue **queries** to the agent system (as a user or environment), influencing what is stored in memory and how retrieval is biased.
- The attacker aims to inject *malicious or misleading episodes and rules* that will later be retrieved and used in planning.
- The attacker may be *stealthy*, balancing attack success with low defense detection to avoid obvious anomalies.
- The defender controls the memory-defense layer (integrity tags, semantic validation, cross-agent checks) but cannot modify the base LLM weights.

6 Phased Execution Roadmap

The project is divided into eleven phases. Each phase states a clear objective, concrete tasks, and deliverables. The goal is to keep the roadmap feasible under deadline pressure, while still supporting the research questions.

Phase 0 – Alignment and Scoping

Objective: Align expectations with advisor; freeze problem scope and success criteria.

Tasks

0.1. Present the consolidated problem statement, RQs, and objectives from this document.

0.2. Confirm target venue type (security / AI systems) and approximate submission window.

0.3. Agree on minimal required experiments (number of scenarios, attacks, and defense configurations).

0.4. Record advisor preferences on emphasis: security vs energy vs multi-agent aspects.

Deliverable

- One-page internal note with agreed RQs, objectives, and scope boundaries.

Phase 1 – Literature Consolidation and Positioning

Objective: Establish a strong conceptual base and clear positioning in the literature.

Tasks

1.1. Select 10–15 papers across three themes: (i) memory attacks on agents, (ii) memory architectures for LLM agents, and (iii) UAV energy-aware planning and security.

1.2. For each paper, fill a structured "paper card" with problem, method, key results, limitations, and relevance.

1.3. Update the proposal and paper's Introduction and Related Work sections to:

   - clearly state the gap: lack of analysis of memory poisoning in energy-constrained multi-UAV LLM systems;
   - position this work relative to MINJA-style attacks and collaborative memory frameworks.

1.4. Identify 2–3 closest works to serve as baselines / foil for the evaluation narrative.

Deliverables

- Updated Introduction and Related Work in the main LaTeX paper.
- Literature notebook (LaTeX or Word) with all paper cards.

Phase 2 – Architecture and Design Freeze

Objective: Finalize the architecture and interfaces before heavy implementation.

Tasks

2.1. Enumerate all agents and their responsibilities: Supervisor, Drone 1 Worker, Drone 2 Worker, and any optional Memory Manager helper.

2.2. Define all tool interfaces explicitly:

- MAVSDK tools: arm, takeoff, goto, land, hold, etc.
- Perception tools: scene queries, detection history.
- Sensor tools: SOC, GPS, distance to home.
- Memory tools: write/query episodic, write/query semantic.

2.3. Define data schemas:

- Episodic entries: $(t, drone\_id, state, action, outcome, mission\_id, MAC)$.
- Semantic rules: $(rule\_id, type, params, support\_episodes, confidence, timestamps)$.

2.4. Sketch the energy model and derive which logs are needed (distance, velocity, time, SOC proxy).

2.5. Draw an architecture diagram for the paper, showing agents, memory, tools, and PX4 SITL.

Deliverables

- Short design document (1–2 pages) with schemas and interfaces.
- Initial architecture figure integrated into the paper.

Phase 3 – Baseline Multi-Agent System (No Attacks, Minimal Defenses)

Objective: Implement a working multi-agent LLM controller for two drones, without attacks or complex defenses.

Tasks

3.1. Implement a single-drone baseline:

- Simple agent or script using MAVSDK to execute waypoint missions from natural-language or structured commands.

3.2. Extend to two drones with a Supervisor and two Workers:

- Supervisor decomposes a mission (e.g., coverage of two regions) into sub-tasks for each worker.
- Workers execute tasks using MAVSDK tools and perception/sensor APIs.

3.3. Implement structured logging of missions (positions, SOC, actions, outcomes) to disk.

3.4. Validate in SITL with several mission runs to ensure stability and basic correctness.

Deliverables

- Reproducible baseline runs for a simple mission scenario (e.g., two-region search).
- Initial System Model section in the paper.

Phase 4 – Memory Layer Implementation

Objective: Implement episodic and semantic memory and integrate them into the decision loop.

Tasks

4.1. Implement the episodic store (e.g., SQLite or Postgres):
   - Write APIs for inserting and querying episodes.
   - Integrate hooks into Supervisor and Workers to write episodes at key decision points.

4.2. Implement the semantic store:
   - Define rule types for hazards, no-fly zones, and energy constraints.
   - Implement operations to add, update, and query semantic rules.

4.3. Integrate retrieval:
   - On each planning step, retrieve relevant episodes and rules (e.g., nearby hazards, recent low-SOC events) to condition agent decisions.

4.4. Add basic instrumentation to log when and how memory entries are used in decisions (for later metrics).

Deliverables

- Working episodic and semantic memory subsystems exercised in baseline missions.
- Updated Architecture / Memory section in the paper.

Phase 5 – Attack Harness (Query-Only Memory Poisoning)

Objective: Implement controllable attacks targeting episodic and semantic memory via query-only interaction.

Tasks

5.1. Design attack scenarios consistent with the threat model:
   - Query patterns that cause the agent to log fake or misleading episodes (e.g., hallucinated obstacles, low-SOC events).
   - Interactions that lead to insertion of malicious semantic rules (e.g., altered hazard maps or energy thresholds).

5.2. Implement attack scripts / functions that:

    - Trigger crafted queries or tasks that result in poisoned episodic entries.
    - Insert or bias the creation of malicious semantic rules through normal agent workflows.

5.3. Tag poisoned entries internally (for ground truth), while keeping the agent unaware of which entries are malicious.

5.4. Implement configuration flags for conditions:

    - Baseline (no attack).
    - Attack-only (defenses disabled).
    - Attack + defenses (Phase 6).

Deliverables

- Attack harness module with a small set of reproducible attack scenarios.
- Documentation of each attack: mechanism, target memory type, expected effect.

Phase 6 – Defense Layer around Memory

Objective: Implement integrity and validation mechanisms to protect memory and enable attack detection/mitigation.

Tasks

6.1. Episodic integrity:

    - Add MACs or signatures to episodic entries at write time.
    - Verify integrity on read; reject or quarantine tampered entries.

6.2. Semantic validation:

    - Require at least k supporting episodes before accepting a new hazard or energy rule.
    - Implement simple conflict detection (e.g., rule contradicts the majority of recent episodes).

6.3. Cross-agent checks:

    - For critical updates (e.g., new hazard region), require corroboration from both drones or diverse evidence.

6.4. Integrate defense decisions into the agents' workflows (e.g., fallback when a rule is rejected).

6.5. Add detailed logging of defense triggers and decisions for later analysis.

Deliverables

- Fully integrated defense layer that can be toggled on/off.
- Defense description and threat model refinement in the paper.

Phase 7 – Experimental Design

Objective: Specify experiments in detail before running them.

Tasks

7.1. Choose 1–2 mission scenarios (e.g., two-region coverage, corridor patrol) with clear success criteria.

7.2. For each scenario, define experimental conditions:

- Baseline (no attack, no defenses).
- Attack-only (attacks enabled, defenses off).
- Attack + defenses (attacks enabled, defenses on).

7.3. Decide on the number of runs per condition (e.g., 10–20) and any randomized initial conditions.

7.4. Map metrics to logs and scripts:

- Mission success rate, safety violations, distance, time.
- Simulated energy usage (energy/km, final SOC).
- Poisoned-use rate (fraction of decisions influenced by poisoned entries).
- Detection and rejection rates by defenses.
- Overhead metrics for the multi-agent security tax: latency per decision, extra tokens, added energy.

7.5. Finalize which tables and figures will appear in the paper (e.g., one summary table, 3–4 key plots).

Deliverables

- Short experimental plan document with scenarios, conditions, metrics, and run counts.

Phase 8 – Evaluation Campaign

Objective: Run experiments, collect clean data, and ensure reproducibility.

Tasks

8.1. Implement automation scripts to launch missions with specific configurations and log outputs into structured files.

8.2. Run a pilot batch of experiments to validate instrumentation and fix bugs.

8.3. Execute the full evaluation campaign according to the experimental plan.

8.4. Store results in a well-organized directory structure (by scenario and condition).

8.5. Perform sanity checks: ranges, missing values, obvious outliers.

Deliverables

- Complete set of logs and summary CSV/JSON files for all runs.

Phase 9 – Analysis, Figures, and Writing

Objective: Turn raw data into tables, figures, and narrative for the paper.

Tasks

9.1. Compute descriptive statistics for all metrics and conditions.

9.2. Generate plots to replace placeholders in the LaTeX paper:
   - Success rates, safety violations.
   - Energy/km and SOC trajectories.
   - Attack impact and defense effectiveness.
   - Overhead plots (latency, tokens, extra energy).

9.3. Fill the Experimental Results section:
   - Replace placeholder numbers in tables.
   - Insert final plots with captions.
   - Write case studies for at least one episodic and one semantic attack scenario.

9.4. Update Discussion and Conclusion with empirical insights and the derived design principles (RQ3).

Deliverables

- Near-final paper draft with real results and polished figures.

Phase 10 – Advisor Iteration and Submission

Objective: Polish the paper to submission quality.

Tasks

10.1. Send the near-final draft to advisor and incorporate feedback over 1–3 iterations.

10.2. Perform a final quality pass on writing, figures, and references.

10.3. Adapt LaTeX to the exact conference template, if needed.

10.4. Upload the final submission and archive the exact submitted version, along with code and scripts (if allowed).

Deliverables

- Submitted paper and archived project (code, configs, and analysis scripts).

Progress Tracking Table

The table below is for tracking progress. Status can be updated manually (Not started / In progress / Done).

Phase	**Status**	**Notes**
Phase 0 – Alignment & Scoping		
Phase 1 – Literature Consolidation		
Phase 2 – Architecture & Design Freeze		
Phase 3 – Baseline Multi-Agent System		
Phase 4 – Memory Layer Implementation		
Phase 5 – Attack Harness		
Phase 6 – Defense Layer		
Phase 7 – Experimental Design		
Phase 8 – Evaluation Campaign		
Phase 9 – Analysis, Figures, & Writing		
Phase 10 – Advisor Iteration & Submission		