UAVLnQ: An Architecture for Security Analysis of Cyber-Physical Network Behavior in UAV Swarms

Abdelrahman Yousef, Chinya Tsai, Nikita Nilesh Mistry, Maria Méndez Real, Guy Gogniat

▶ **To cite this version:**

Abdelrahman Yousef, Chinya Tsai, Nikita Nilesh Mistry, Maria Méndez Real, Guy Gogniat. UAVLnQ: An Architecture for Security Analysis of Cyber-Physical Network Behavior in UAV Swarms. DASIP: Workshop on Design and Architectures for Signal and Image Processing, In conjonction with HIPEAC, Jan 2026, Krakovie, Poland. ⟨hal-05537630⟩

HAL Id: hal-05537630

https://hal.science/hal-05537630v1

Submitted on 5 Mar 2026

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons CC BY-NC 4.0 - Attribution - Non-commercial use - International License

UAVLnQ: An Architecture for Security Analysis of Cyber-Physical Network Behavior in UAV Swarms

Abdelrahman Yousef[1], Chinya Tsai[1], Nikita Nilesh Mistry[2], Maria Méndez Real[3], and Guy Gogniat[3]

[1] Université Bretagne Sud, Lorient, France
[2] Department of Computer Science & Engineering, College of Engineering, Qatar University, Doha, Qatar
[3] UMR 6285, Lab-STICC, Université Bretagne-Sud, Lorient, France

Abstract. The design and validation of security architectures for Unmanned Aerial Vehicle (UAV) swarms, such as on-board Intrusion Detection Systems (IDS), require simulation platforms that can rapidly prototype complex cyber-physical scenarios. These platforms must simultaneously capture accurate flight dynamics and realistic network behavior while remaining scalable and reproducible. This paper introduces Unmanned Aerial Vehicle Link and Queue (UAVLnQ), an open-source architecture explicitly designed to address these challenges. It integrates ArduPilot Software-in-the-Loop (SITL) with ns-3 through a ZMQ middleware that guarantees synchronization between mobility control and packet-level events. UAVLnQ features an autopilot mission control system for defining and executing complex swarm behaviors. We demonstrate its capabilities in a leader-follower formation under representative cyber-attacks—including command injection, spoofing, and denial of service—providing a validated environment for the development and testing of novel security and resilience solutions. UAVLnQ thus establishes itself as a reproducible and extensible foundation for a new generation of UAV swarm networking and cyber-defense research.

Keywords: Unmanned Aerial Vehicles · ns-3 · ArduPilot · MAVLink · UAV Swarm · UAV Attacks.

1 Introduction

Unmanned Aerial Vehicle (UAV) swarms are increasingly characterized by autonomous and cooperative operations in which mission success depends on reliable and resilient wireless mesh networking [**? ?**]. Such swarms are now deployed in safety-critical applications including emergency response, surveillance, and infrastructure inspection, where communication failures or malicious interference can rapidly propagate into unsafe physical behavior or mission failure. However, this same reliance on wireless communication introduces a substantial attack surface; adversarial actors may exploit vulnerabilities through command injection, GPS spoofing, or denial-of-service (DoS) attacks, leading to loss of control, trajectory deviation, or forced termination of swarm missions [**? ?**].

As swarm coordination logic and autonomy grow in complexity, there is an increasing need for simulation platforms that accurately model the cyber–physical coupling between network-layer events and UAVs [? ?]. In particular, the validation of Intrusion Detection Systems (IDS) depends on preserving the causal relationship between network anomalies and physical responses, as decoupled simulations can lead to misleading assessments of system resilience [?].
Direct experimentation with cyber-physical UAV attacks is often infeasible due to safety concerns, regulatory constraints, and legal restrictions on jamming and spoofing in open airspace [? ?]. Consequently, simulation environments have become the primary means for developing and validating UAV security mechanisms. However, many existing simulation frameworks fail to preserve the fidelity required for cyber-physical security analysis [?]. Robotics-centric simulators such as Gazebo abstract communication into idealized inter-process messaging, preventing the evaluation of protocol-level exploits [?]. Hybrid co-simulation frameworks, most notably FlyNetSim [? ?], integrate ArduPilot with ns-3 but often bypass the simulated network for mobility or position updates to improve scalability. This decoupling can produce a false sense of resilience: a simulated denial-of-service attack may saturate the network, yet UAVs continue executing their missions because control inputs are delivered out-of-band.
Similarly, lightweight frameworks such as DroNS-3 emphasize ease of use and high-level mission logic but rely on abstractions that obscure low-level packet dynamics and timing behavior. While such tools are valuable for algorithm prototyping, they are ill-suited for validating IDS mechanisms where packet-level visibility and causal cyber-physical effects are essential [? ?]. As a result, existing platforms struggle to support reproducible security experiments in which network-layer attacks induce realistic and observable physical consequences. This work introduces UAVLnQ, an open-source cyber-physical co-simulation architecture explicitly designed to close this validity gap. Unlike existing frameworks, UAVLnQ enforces a strict coupling between communication and mobility by ensuring that all MAVLink control, telemetry, and inter-UAV coordination traffic traverses the simulated wireless network. Any network degradation or malicious manipulation therefore directly influences UAV flight behavior.

The primary objectives of this work are fourfold:

- To accurately model UAV flight dynamics and mission execution using ArduPilot Software-in-the-Loop (SITL) while preserving native MAVLink communication semantics;
- To simulate realistic wireless networking conditions in which latency, packet loss, and adversarial traffic causally propagate into UAV behavior;
- To support configurable multi-UAV swarm topologies with inter-UAV communication and coordination; and
- To provide a reproducible adversarial experimentation framework for UAV cybersecurity and Intrusion Detection System research.

UAVLnQ achieves these objectives by tightly integrating ArduPilot SITL with ns-3 through a lightweight ZeroMQ middleware that preserves the MAVLink

communication pipeline while enabling low-latency, bidirectional message exchange. The framework produces synchronized packet-level network traces alongside timestamped ArduPilot telemetry logs, enabling the generation of high-fidelity cyber-physical datasets suitable for IDS development and security evaluation. UAVLnQ is fully open source and publicly available [**?**].
We validate the proposed architecture using a leader–follower swarm topology in which the leader UAV acts as a swarm controller, issuing MAVLink commands to follower UAVs over the simulated network. Scalability is evaluated up to 20 UAVs with stable resource usage and end-to-end latency below 100 ms. Representative cyber-attack scenarios including command injection, spoofing, and DoS are introduced to demonstrate how network-layer exploits propagate into observable flight anomalies. These results establish UAVLnQ as a reliable and extensible foundation for UAV swarm networking and cyber-defense research.

2 Related Work

UAV simulation environments are broadly categorized into robotics-centric, network-centric, and integrated co-simulation architectures.

2.1 Robotics and Digital Twin Simulators

General-purpose robotics simulators are frequently used to model UAVs because they provide the necessary physics engines to replicate flight dynamics and aerodynamic forces. Platforms such as Gazebo [**?**] and AirSim [**?**] excel at high-fidelity physical modeling and photorealistic rendering. However, they abstract communication into idealized inter-process messaging, neglecting stochastic network phenomena—such as packet loss, jitter, and protocol overhead—which are foundational for effective security research. Recent Digital Twin implementations, notably GazeboNS3 [**?**], extend these platforms by integrating battery models and computational constraints. GazeboNS3 is highly effective for optimizing the energy profile of specific hardware, yet its dependency on full physics rendering limits scalability. Benchmarks indicate that real-time performance in such systems drops noticeably as node counts rise above ten. Additionally, these systems often lack native modules for adversarial packet injection, requiring extensive modification to support cybersecurity workflows. Trace-based tools like DroneNet-Sim [**?**] offer an alternative by replaying recorded mobility traces. While computationally efficient, trace-based systems cannot react dynamically to attacks—a drone will not stop when a command is jammed if it is following a pre-recorded path—rendering them unsuitable for active defense validation.

2.2 Network-Centric Simulators

Discrete-event network simulators, such as ns-3, OMNeT++ [**?**], and OPNET, excel at modeling the lower layers of the protocol stack, offering granular control over channel interference, propagation loss, and routing logic [**?**]. However, these platforms treat mobile nodes as abstract entities governed by simplified kinematic models (e.g., Random Waypoint or Gauss-Markov) rather than dynamic physical bodies [**?**]. They lack the aerodynamic engines necessary to simulate inertia, wind resistance, or thrust-to-weight ratios. Consequently, standalone

network simulators cannot capture the cyber-physical feedback loop essential for security validation; they cannot model scenarios where network latency induces mechanical instability or where a kinetic maneuver inadvertently disrupts the antenna link budget. While recent Python-based extensions like UavNetSim-v1 [?] attempt to introduce mobility features, they lack the real-time synchronization with flight controllers required to test active autopilot defense mechanisms.

2.3 Integrated Co-Simulation Architectures

Co-simulation frameworks link distinct simulators to address networking fidelity, though often with significant trade-offs. FlyNetSim [?] pioneered the ArduPilot/ns-3 integration via middleware. However, it's reported scalability is measured without accounting for full intra-swarm mesh communication, making its statistics unsuitable for rigorous cybersecurity validation where coordinated, network-dependent attacks must propagate through the simulated channel.
While frameworks such as ROS-NetSim [?], FANET-Sim [?] and Cornet [?] provide robust co-simulation capabilities, their dependence on the Robot Operating System (ROS) [?] introduces substantial computational overhead that constrains scalability for large swarms. In contrast, DroNS-3 offers a streamlined interface between ArduPilot and ns-3 tailored for educational use and single-agent logic. However, its reliance on the DroneKit [?] API abstracts critical low-level packet dynamics, obscuring the data granularity necessary for validating Intrusion Detection Systems (IDS). Other approaches, such as CUSCUS [?], leverage Linux containers to bridge simulators but remain proprietary and closed-source. UAVLnQ improves upon these state-of-the-art methods by implementing a ZMQ middleware, which guarantees tight cyber-physical coupling without the resource penalties of ROS or heavy physics engines, thereby optimizing the system for scalable swarm simulation and adversarial injection.

3 UAVLnQ Framework

The UAVLnQ framework integrates several core components into a single simulation environment. At its foundation lies a flight simulator that provides realistic Unmanned Aerial Vehicle (UAV) dynamics and mission control logic, coupled with a network simulator that models communication links at the packet level. A middleware layer connects these two domains, forming a closed-loop architecture for the analysis.

3.1 Core Components

To achieve realistic flight dynamics and mission control, UAVLnQ integrates ArduPilot SITL [?] UAVs, which are visualized in 3D via Gazebo [?] or in 2D via QGroundControl [?], and the DroneKit API [?] mission planning and controller for high-level mission logic, while ns-3 [?], a discrete-event network simulator, models packet level communication across the entire swarm topology including inter-UAV links and ground control station connectivity.
Since these components operate as separate processes, we connect them using a ZMQ [?] middleware that supports bidirectional message exchange. This

publish-subscribe architecture creates end-to-end data paths for mission commands, telemetry data, and network packets.

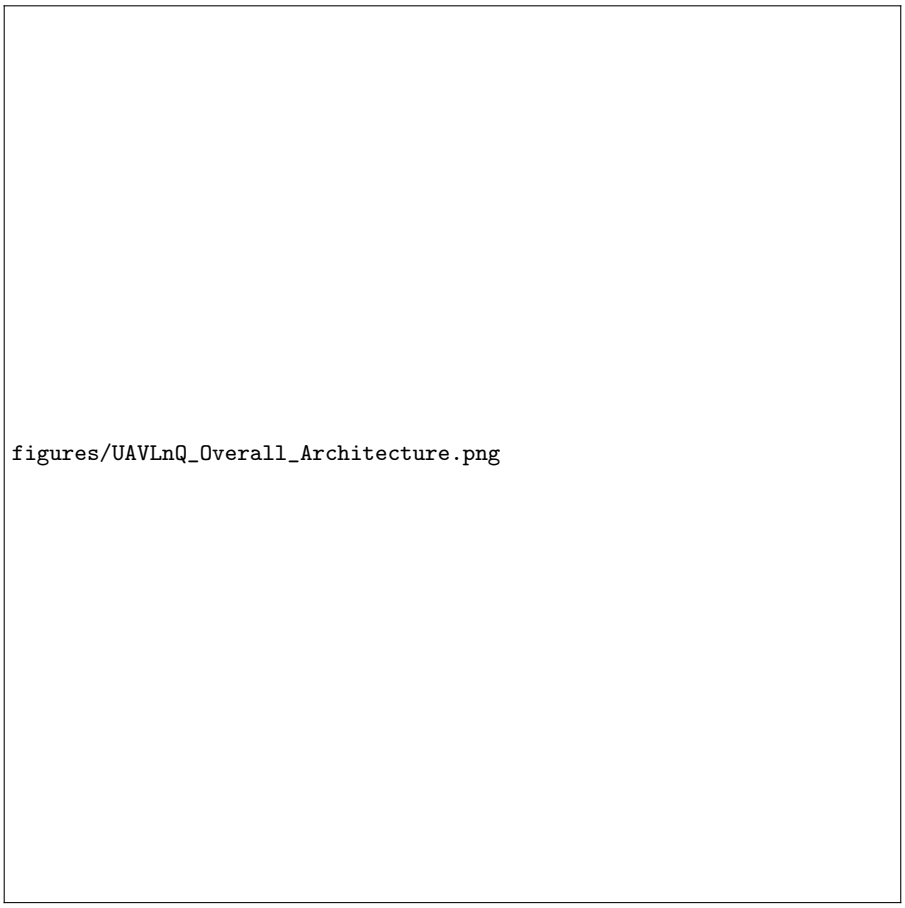


Fig. 1: UAVLnQ Framework

3.2 Overall Architecture

Figure 1 illustrates the overall architecture of the UAVLnQ framework. The mission control logic, implemented in Python, serves as the central coordinator, managing connections and mission execution for all UAV instances while logging telemetry data. Communication is handled via two primary ZMQ sockets (ports 5555 and 5556) that bridge MAVLink traffic between ArduPilot/mavutil components and the ns-3 simulation which ensures that MAVLink messages from ArduPilot reach ns-3 for network propagation, while packets generated in ns-3 are delivered back to the corresponding ArduPilot instances for execution. Unlike approaches based on high-overhead visualizers such as AirSim, UAVLnQ employs a ZMQ bridge that acts as a low-latency pass-through between simulators. Through direct one-to-one mapping between UAV instances and ns-3 nodes,

its design leverages ns-3's inherent capability to handle complex topologies while capturing comprehensive network traffic traces of the entire swarm topology. The multi-UAV simulation is configured via a single JSON file that defines per-UAV parameters including DroneKit connection endpoints, MAVLink communication and parser ports, QGroundControl visualization ports and the path of the ns-3 network simulation script, allowing straightforward swarm scaling by adding configuration entries. This architecture enables UAVLnQ to be customizable and scale to large swarm systems.

Table 1: UAVLnQ Commands

Command	**Description**
MoveToWaypoint	Move the UAV to specified local coordinates.
Sleep	Pause execution of commands for a specified duration.
ReturnHome	Return to the launch location.
Land	Land at the current UAV position.

4 UAVLnQ Autopilot

The autopilot architecture shown in Figure 2 comprises five modular components working in concert to manage swarm operations. The DroneCommander class coordinates all UAV connections by reading the JSON configuration file to obtain per-UAV connection parameters, establishing and managing parallel connections via DroneKit while verifying armable state before mission initiation. It publishes MAVLink messages to ns-3 through ZMQ, launches the network simulator, and monitors mission files for dynamic command updates. Individual DynamicMissionController instances manage per-UAV command execution, where commands are defined through a base class interface that issues instructions to ArduPilot SITL via the DroneKit API. Table 1 provides an overview of available UAV commands. The DataLogger class captures telemetry data and stores it in timestamped CSV files, while the MAVLink Parser bridges network traffic between ns-3 and autopilot instances. This modular architecture allows rapid prototyping of swarm missions and cyber attack scenarios with low implementation complexity and provides the flexibility to dynamically modify mission behavior during execution.

5 Evaluation Analysis

To assess UAVLnQ's scalability, computational efficiency, and communication latency characteristics, we evaluate the framework across different swarm sizes by executing standardized mission scenarios. All experiments were conducted on an AMD Ryzen 5 7520U (4 cores, 8 threads, 4.4 GHz max) with 16 GB RAM running Ubuntu 24.04 LTS. The evaluation quantifies computational overhead as swarm size increases and characterizes the ZMQ-based middleware latency for both position updates and command propagation between flight simulation and network modeling.

5.1 Computational Overhead and Scalability

CPU utilization and memory consumption was evaluated across swarm sizes of 3, 10, and 20 UAVs executing 300-second missions. Each configuration was tested through 10 independent runs to ensure statistical reliability, with results averaged across repetitions. Table 2 shows that CPU utilization scales approximately linearly with swarm size, increasing from 29.8% (3 UAVs) to 61.8% (10

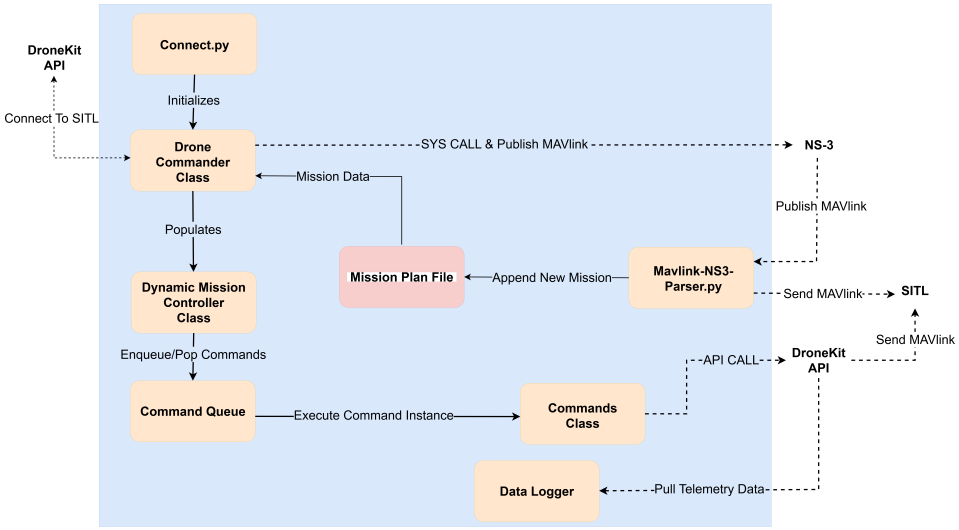


Fig. 2: UAVLnQ Autopilot Structure

UAVs) and 86.2% (20 UAVs), indicating CPU becomes the primary bottleneck. Memory consumption remains stable at approximately 9.5-10 GB across all configurations, dominated by fixed overhead from ns-3 and QGroundControl rather than per-UAV costs. The framework maintains efficient execution up to 10 UAVs with comfortable headroom (62% CPU), while 20 UAVs approaches saturation at 86% CPU.

Table 2: UAVLnQ Computational Overhead

Number of UAVs	**Avg. CPU (%)**	**Avg. Memory (MB)**
3	29.8	9,565
10	61.8	9,817
20	86.2	9,947

5.2 Communication Latency Analysis

To characterize middleware overhead, we measured latencies for two data paths across swarms of 1 to 10 UAVs. Position update latency captures the delay from GPS_RAW_INT messages generation in ArduPilot SITL to mobility model synchronization in ns-3. Meanwhile, end-to-end command latency measures the delay from the moment a MAVlink packet is generated and transmitted from ns-3 to the point where it is queued as a command in the target follower's autopilot. Figure 3 presents both latency metrics as swarm size increases. Position update latency averages 84.3 ms overall, scaling from 76.5 ms for 1 UAV to 84.4 ms for 10 UAVs, representing a modest 10.4% increase. End-to-end command latency exhibits similar behavior, averaging 83.3 ms overall and increasing from 79.8 ms to 83.3 ms across the same range, a 4.4% increase. Both metrics remain consistently below 100 ms across all tested swarm sizes.

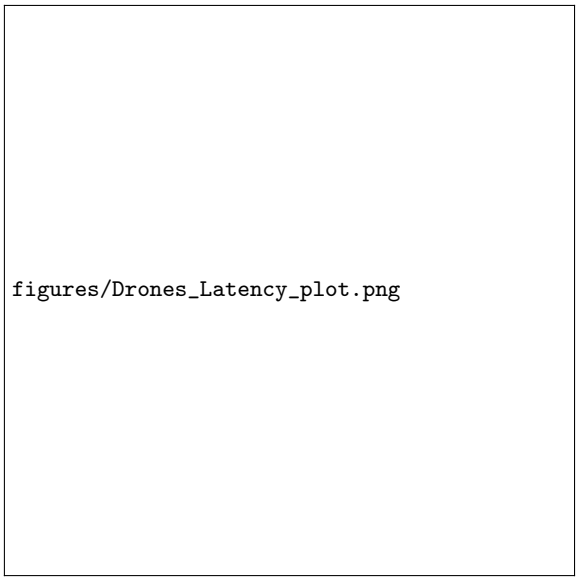


Fig. 3: Communication latency vs. swarm size. Position update delay and end-to-end command latency.

6 Use Cases

We demonstrate UAVLnQ's capabilities through two representative scenarios: (i) a baseline mission operation under nominal conditions, and (ii) the same topology subjected to cyberattacks. These scenarios validate the framework's ability to accurately reproduce UAV mission execution while systematically modeling adversarial interference within a controlled and reproducible environment, generating correlated datasets.

6.1 Case I: Normal Mission Operation

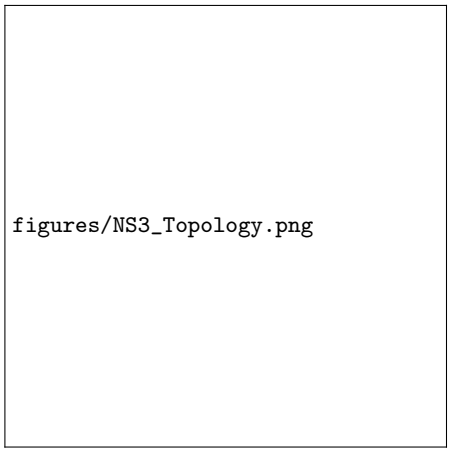
The first use case comprises three UAV nodes in a fully connected mesh network shown in Figure 4a. Node mobility is synchronized with ArduPilot Software-in-the-Loop (SITL) via `GPS_RAW_INT` messages over ZMQ, while communication follows the MAVLink protocol [?]: the leader sends `MISSION_ITEM` commands to followers, and all UAVs broadcast telemetry for swarm-wide awareness.

Figure 4b shows the mission flight path of each UAV, as the leader executes its mission and dispatches waypoints to followers at scheduled intervals. Upon leader landing, all followers execute Return-to-Launch for coordinated mission termination.

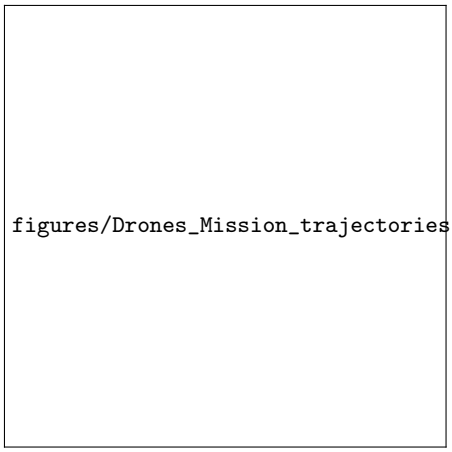
Generated Dataset. Table 3 summarizes the dataset generated from the baseline mission. Each UAV produces telemetry logs sampled at 0.5-second intervals, while ns-3 captures complete packet level network traces.

Table 3: Case I Dataset Statistics

Metric	**Value**
Mission Duration	200 seconds
Telemetry CSV Samples per UAV	400 samples
Total Network Packets	25,000 packets



(a) ns-3 Topology



(b) Flight Trajectories

Fig. 4: Baseline mission: network topology and flight paths.

6.2 Case II: Normal Mission Operation Under Cyber Attacks

Extending the first use case. A malicious node was introduced in the same network topology to intercept unencrypted MAVLink traffic and inject crafted packets to both UAVs and ground station to evaluate the adversarial behaviors and their effects on flight dynamics.

6.3 Attack Categories

UAVLnQ supports prototyping of several classes of UAV attacks using different MAVlink messages. We validated four representative categories frequently considered in UAV security research:

- **Command Injection:** Injected spoofed commands including `MISSION_ITEM` to modify mission waypoints, `MAV_CMD_COMPONENT_ARM_DISARM` for forced disarm, `MAV_CMD_DO_FLIGHTTERMINATION` for full flight termination, and `MAV_CMD_DO_CHANGE_SPEED` to change flight speed.
- **Spoofing:** Falsified `GPS_RAW_INT` packets distorting localization and manipulated `BATTERY_STATUS` triggering unintended failsafes.
- **Navigation Hijacking:** Spoofed `MAV_CMD_DO_SET_HOME` commands to alter home position, thereby redirecting the Return-to-Launch (RTL) behavior.
- **Denial of Service (DoS) Attacks:** Flood the communication channels with malicious traffic, including `HEARTBEAT` flooding and ghost drone flooding (simultaneous injection of `GLOBAL_POSITION_INT`, `HEARTBEAT`, `SYS_STATUS`, and `GPS_RAW_INT` to create fictitious UAVs).

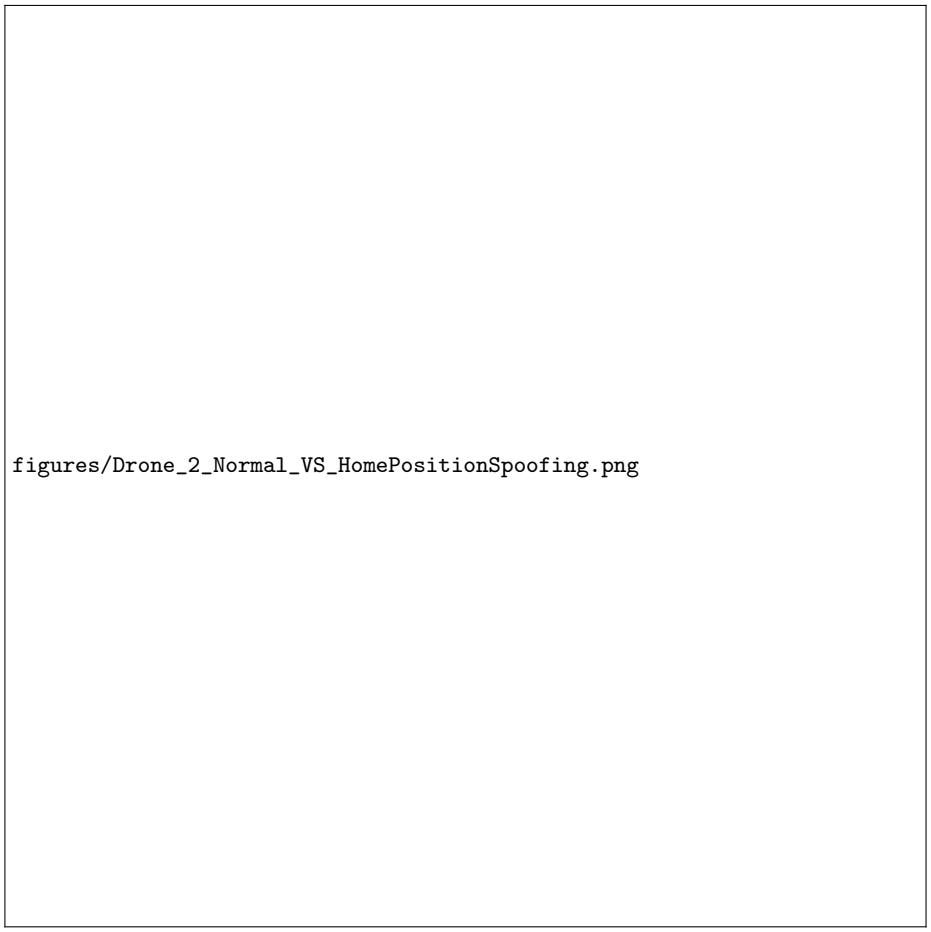


Fig. 5: 2nd UAV trajectory comparison: Normal mission vs Home Position Hijacking Attack

6.4 Scenario I: Navigation Hijacking

The attacker starts injecting spoofed `MAV_CMD_DO_SET_HOME` commands into the simulation, targeting 2nd UAV and repeatedly redirecting its home position to a point 124 m far from the original launch site. The attack function reschedules itself every second, creating a persistent home position spoofing flood. Figure 5 illustrates the resulting trajectory divergence: under normal conditions, the UAV completes its waypoints and returns to the original home location; under attack, it diverts to the hijacked coordinates upon receiving the return-to-launch command. The redirected path increases the total flight distance from 247 m to 275 m, while the mission duration decreases to 177 s because the spoofed home location is closer to the final waypoint. Table 4 summarizes the corresponding dataset characteristics.

Table 4: Dataset Comparison for 2nd UAV: Normal vs Home Position Hijacking Attack

Metric	**Normal**	**Attack**
Mission Duration	201 seconds	177 seconds
Path Distance	247 meters	275 meters
End Position Deviation	0 meter	124 meters
Avg telemetry samples per UAV	400 samples	353 samples
Total Network Packets	25,000 packets	27,332 packets

6.5 Scenario II: Operational Efficiency Degradation

The third scenario demonstrates operational degradation through speed manipulation. The attacker injects `MAV_CMD_DO_CHANGE_SPEED` commands every 5 seconds, reducing second UAV ground speed to 0.5 m/s. Table 5 shows mission duration extended from 201 seconds to 303 seconds, while average ground speed decreased from 1.25 m/s to 0.80 m/s. Table 5 also compares dataset characteristics.

Table 5: Dataset Comparison for the 2nd UAV: Normal vs Speed Manipulation Attack

Metric	**Normal**	**Attack**
Mission Duration	201 seconds	303 seconds
Avg Ground Speed	1.25 m/s	0.80 m/s
Avg telemetry samples per UAV	400 samples	605 samples
Total Network Packets	25,000 packets	44,477 packets

7 Conclusion

UAV swarms are becoming essential in emergency response, logistics, and security, yet their development is limited by the lack of simulation platforms that simultaneously capture realistic flight dynamics, packet-level network behavior, and cyber-physical attack propagation. UAVLnQ addresses this gap by integrating ArduPilot SITL with ns-3 through a ZMQ middleware, enabling synchronized mobility and communication modeling. As a general-purpose and extensible simulator, UAVLnQ supports rapid prototyping of swarm missions and attack scenarios with low implementation complexity. The framework's modular architecture, driven by JSON configuration and command-based mission control, enables straightforward adaptation for research and education and extends beyond multirotors to fixed-wing aircraft and ground vehicles. Our evaluation demonstrated UAVLnQ's computational efficiency and scalability, with stable memory consumption and communication latency remaining consistently below 100 ms for both position updates and command propagation. We validated the framework's cyber-physical testbed capabilities through two representative attack scenarios—navigation hijacking and operational efficiency degradation to

demonstrate observable consequences in both network traces and flight telemetry. Each scenario generates correlated datasets combining pcap network captures and CSV telemetry logs, providing data that can be used for instance to build and test IDS models. Compared to existing solutions, UAVLnQ provides an efficient platform for simulating realistic cyber-physical behavior across large UAV swarms. The complete source code and tool is released as open source [**?**] and our hope is that it will serve as a basis for the development of new contributions in this direction. Future work will expand the framework with additional attack models, defensive mechanisms, and more complex swarm topology and coordination strategies, alongside developing lightweight IDS solutions deployable on-board UAVs.

Acknowledgments. This work was carried out as part of the CYBERUS Erasmus Mundus Master's programme. The authors would like to thank the programme for its support.