



# **University of Wales Trinity Saint David**

## **MASTER'S PROJECT REPORT**

### **A Security Framework for Detecting Prompt-Injection-Induced Routing Manipulation in AI-Assisted UAV Swarm Networks**

**Naymul Hasnat**

Student ID: 2432483

**Project submitted as part of the requirements for the award of**

**MSc Cyber Security and Digital Forensics**

Supervisor: Dipo Dunsin

September 2026

# DECLARATION AND CONSENT

## Declaration

I, Naymul Hasnat, declare that I am the sole author of this Master's Project and that all sources and references used in the preparation of this report have been appropriately acknowledged. I confirm that the work presented in this project was undertaken by me and that the completed report is within the prescribed word limit.

I further confirm that this work has not previously been submitted or accepted, in whole or in part, for the award of any other degree or academic qualification.

**Signature:** \_\_\_\_Naymul Hasnat\_\_\_\_

**Date:** \_\_\_\_24/09/2026\_\_\_\_

## Form of Consent

I, Naymul Hasnat, hereby consent that my Master's Project, submitted in candidature for the award of MSc Cyber Security and Digital Forensics, if successful, may be made available for inter-library loan or photocopying, subject to copyright regulations. I also consent that the title and abstract of this project may be made available to external organisations where appropriate.

**Signature:** \_\_\_\_Naymul Hasnat\_\_\_\_

**Date:** \_\_\_\_24/09/2026\_\_\_\_

## Abstract

The increasing use of artificial intelligence (AI) in unmanned aerial vehicle (UAV) swarm networks introduces new security challenges, particularly when Large Language Models (LLMs) are used to support routing and network-related decisions. One emerging threat is prompt injection, where malicious instructions can manipulate an AI system into generating inappropriate or unsafe routing recommendations. If such recommendations are accepted without sufficient validation, they may influence routing behaviour within a UAV swarm and potentially expose the network to further attacks. This research investigates the effectiveness of a multi-layer security framework for detecting and reducing prompt-injection-induced routing manipulation in AI-assisted UAV swarm networks.

A Python-based simulation environment was developed to represent a UAV swarm operating through dynamic ad hoc/Flying Ad Hoc Network (FANET) communication. A deterministic LLM surrogate was used in the primary experiments to generate repeatable AI-assisted routing decisions without relying on external API availability or model randomness. The proposed framework consists of four defence layers. D1 focuses on detecting potentially malicious prompt inputs, while D2 validates AI-generated routing recommendations against routing and network constraints. D3 monitors UAV and routing behaviour to identify suspicious activity during operation. D4 provides an additional active verification layer for suspicious cases that remain unresolved by the preceding mechanisms. It uses information from other UAVs for corroboration and can initiate further probe-based verification when additional evidence is required.

The framework was evaluated through controlled simulation experiments involving prompt-injection-induced routing manipulation scenarios. The experimental results, represented by the graphs generated from the implementation, were used to examine the contribution of the layered defence configurations and the additional protection provided by D4. The study demonstrates the value of combining LLM security controls with UAV network-level monitoring and active verification. The research contributes a practical, simulation-based framework for investigating the security of AI-assisted UAV routing and provides a basis for further research into adaptive and independently verified AI decision-making in autonomous UAV networks.

# Table of Contents

|                                                          |      |
|----------------------------------------------------------|------|
| DECLARATION AND CONSENT .....                            | ii   |
| Abstract .....                                           | iii  |
| Table of Contents .....                                  | iv   |
| List of Tables .....                                     | viii |
| List of Figures .....                                    | ix   |
| Chapter 1: Introduction .....                            | 1    |
| 1.1 Background .....                                     | 1    |
| 1.2 Problem Statement .....                              | 2    |
| 1.3 Aim of the Study .....                               | 3    |
| 1.4 Research Question.....                               | 3    |
| 1.5 Research Objectives .....                            | 3    |
| 1.6 Proposed Security Framework .....                    | 4    |
| 1.7 Research Method and Scope .....                      | 4    |
| 1.8 Research Gap .....                                   | 5    |
| 1.9 Significance and Structure of the Report .....       | 5    |
| Chapter 2: Literature Review .....                       | 7    |
| 2.1 UAV and Flying Ad Hoc Network Background.....        | 7    |
| 2.2 UAV Routing and Routing Security .....               | 8    |
| 2.3 AI and LLM-Enabled UAV Systems.....                  | 12   |
| 2.4 Prompt Injection and LLM Security .....              | 14   |
| 2.4.1 Prompt-level detection .....                       | 17   |
| 2.5 UAV Intrusion Detection and AI-Based Detection ..... | 18   |
| 2.6 LLM-Agent and UAV Recent Research.....               | 20   |
| 2.7 Critical Comparison of Existing Work.....            | 22   |
| 2.8 Research Gap .....                                   | 26   |
| 2.9 Summary of the Literature Review .....               | 27   |

|                                                                       |    |
|-----------------------------------------------------------------------|----|
| Chapter 3: Research Methodology, System Design and Architecture ..... | 28 |
| 3.1 Chapter Overview .....                                            | 28 |
| 3.2 Research Methodology.....                                         | 28 |
| 3.3 Design Requirements .....                                         | 29 |
| 3.4 Overall System Architecture .....                                 | 30 |
| 3.5 Architectural Components.....                                     | 31 |
| 3.5.1 Ground Control Station .....                                    | 31 |
| 3.5.2 AI-Assisted Routing Engine.....                                 | 32 |
| 3.5.3 Multi-Layer Security Framework .....                            | 32 |
| 3.5.4 UAV Swarm and FANET Layer.....                                  | 33 |
| 3.6 Data Flow and Processing Pipeline.....                            | 33 |
| 3.7 Design of Defence Layers .....                                    | 34 |
| 3.7.1 D1 – Prompt Injection Detection and Input Validation.....       | 34 |
| 3.7.2 D2 – Output Verification and Policy Compliance.....             | 35 |
| 3.7.3 D3 – Behavioural Monitoring and Anomaly Detection .....         | 35 |
| 3.7.4 D4 – Active Verification and Trust Management.....              | 35 |
| 3.8 Chapter Summary.....                                              | 35 |
| Chapter 4: Implementation and Experimental Configuration .....        | 37 |
| 4.1 Chapter Overview .....                                            | 37 |
| 4.2 Software Architecture and Project Structure.....                  | 37 |
| 4.3 Simulation Environment and Network Model .....                    | 43 |
| 4.4 Synthetic Data Generation .....                                   | 43 |
| 4.5 LLM-Assisted Routing Component.....                               | 44 |
| 4.6 D1 Implementation – Prompt Injection Detection .....              | 44 |
| 4.7 D2 Implementation – Routing Output Verification.....              | 44 |
| 4.8 D3 Implementation – Behavioural Anomaly Detection.....            | 45 |
| 4.9 D4 Implementation – Corroboration and Active Verification.....    | 45 |
| 4.10 Attack Scenario Design .....                                     | 46 |

|                                                                               |    |
|-------------------------------------------------------------------------------|----|
| 4.11 Experimental Baselines .....                                             | 46 |
| 4.12 Experimental Procedure .....                                             | 47 |
| 4.13 Attack-Success and Clean-Traffic Measures.....                           | 47 |
| 4.14 Data Storage and Reproducibility .....                                   | 48 |
| 4.15 Chapter Summary.....                                                     | 48 |
| Chapter 5: Experimental Results.....                                          | 49 |
| 5.1 Chapter Overview .....                                                    | 49 |
| 5.2 Experimental Dataset and Evaluation Runs .....                            | 49 |
| 5.3 Primary Attack-Success Results.....                                       | 49 |
| 5.4 Results for Individual Attack Scenarios.....                              | 51 |
| 5.4.1 Routing-Manipulation-Only Scenario.....                                 | 51 |
| 5.4.2 Prompt-Injection-Only Scenario .....                                    | 51 |
| 5.4.3 Combined Scenario .....                                                 | 51 |
| 5.5 Clean-Traffic Results and False Positive Rate .....                       | 51 |
| 5.6 Held-for-review and Probe Dispatch .....                                  | 53 |
| 5.7 Overt versus Stealthy Prompt Injection .....                              | 54 |
| 5.8 Observed Contribution of D4.....                                          | 55 |
| 5.9 Summary of Main Findings .....                                            | 56 |
| Chapter 6: Evaluation and Discussion .....                                    | 57 |
| 6.1 Evaluation Overview.....                                                  | 57 |
| 6.2 Evaluation Against the Research Objectives .....                          | 57 |
| 6.3 Overall Security Effectiveness .....                                      | 57 |
| 6.4 Discussion of D1: Prompt-Injection Detection .....                        | 57 |
| 6.5 Discussion of D2 and D3: Independent Routing and Behavioural Checks ..... | 58 |
| 6.6 Discussion of D4: Active Verification.....                                | 58 |
| 6.7 Overt versus Stealthy Attacks .....                                       | 59 |
| 6.8 Limitations of the Evaluation.....                                        | 59 |
| 6.9 Answer to the Research Question .....                                     | 59 |

|                                                 |    |
|-------------------------------------------------|----|
| 6.10 Chapter Summary.....                       | 60 |
| Chapter 7: Conclusion and Recommendations ..... | 61 |
| 7.1 Conclusion .....                            | 61 |
| 7.2 Recommendations .....                       | 61 |
| 7.3 Suggestions for Further Work .....          | 61 |
| 7.4 Closing Statement .....                     | 62 |
| Chapter 8: Reflection .....                     | 63 |
| References .....                                | 64 |

## List of Tables

|                                                                         |    |
|-------------------------------------------------------------------------|----|
| Table 1 Literature synthesis: study focus and relevance .....           | 23 |
| Table 2 Critical relevance and limitations of the reviewed studies..... | 25 |
| Table 3 Attack Scenarios .....                                          | 46 |
| Table 4 Experimental evaluation configuration .....                     | 49 |
| Table 5 Attack-success results from the primary evaluation sweep.....   | 50 |
| Table 6 False positive results on clean traffic .....                   | 52 |
| Table 7 Held-for-review rate and probe-dispatch rates .....             | 53 |
| Table 8 Overt-versus-stealthy prompt-injection results .....            | 54 |
| Table 9 Summary of principal primary-sweep results .....                | 56 |

## List of Figures

|                                                                                                             |    |
|-------------------------------------------------------------------------------------------------------------|----|
| Figure 1 General architecture of a Flying Ad Hoc Network (FANET). Source: [1].....                          | 7  |
| Figure 2 Classification of FANET routing approaches. Source: [1] .....                                      | 10 |
| Figure 3 Overview of UAV–foundation-model integration and the path towards agentic UAVs. Source: [14] ..... | 13 |
| Figure 4 Illustrative example of normal and jailbreak interaction modes. Source: [15].                      | 16 |
| Figure 5 UAV swarm intrusion-detection architecture based on the cyber-physical feature. Source: [9].....   | 19 |
| Figure 6 LLM-centric agentic UAV swarm architecture. Source: [21].....                                      | 21 |
| Figure 7 Overall System Architecture of the Proposed Security Framework.....                                | 31 |
| Figure 8 Sequence of Routing Request, Multi-Layer Security Verification and Telemetry Feedback .....        | 34 |
| Figure 9 main.py .....                                                                                      | 37 |
| Figure 10 swarm_simulator.py.....                                                                           | 38 |
| Figure 11 routing_protocol.py.....                                                                          | 39 |
| Figure 12 prompt_injection.py.....                                                                          | 39 |
| Figure 13 llm_agent.py .....                                                                                | 40 |
| Figure 14 detector.py.....                                                                                  | 41 |
| Figure 15 trust_manager.py.....                                                                             | 41 |
| Figure 16 verification_agent.py .....                                                                       | 42 |
| Figure 17 train_detector.py .....                                                                           | 43 |
| Figure 18 Attack success rate under the combined attack scenario .....                                      | 50 |
| Figure 19 False positive rate on clean traffic across the three defence configurations...                   | 52 |
| Figure 20 Attack success rate for overt and stealthy prompt injection .....                                 | 55 |

# Chapter 1: Introduction

## 1.1 Background

Unmanned Aerial Vehicles (UAVs) are increasingly used in applications that require flexible communication, rapid deployment and coordinated operation. When multiple UAVs communicate directly with one another and cooperate to perform a common task, they can form a Flying Ad Hoc Network (FANET). Unlike conventional wireless networks, FANETs operate with highly mobile nodes and frequently changing network topologies. Maintaining reliable communication and selecting appropriate routes between UAVs are therefore challenging, particularly because mobility, dynamic topology, limited communication range and link instability can affect routing performance [1].

The dynamic characteristics of UAV swarm networks have encouraged the development of adaptive networking approaches. Chen et al. proposed an adaptive drone swarm network framework designed to improve communication flexibility and adaptability under changing operating conditions [2]. Research on autonomous and intelligent UAV swarms has also highlighted the growing role of automated coordination, intelligent decision-making and network management in distributed aerial systems [3].

Intelligent techniques have also been investigated from a network-security perspective. Zhang et al. used deep reinforcement learning to identify attack sequences capable of decomposing a large UAV swarm communication network, showing that the connectivity structure of a FANET can itself become a target of intelligent attack [4]. Leonov examined bio-inspired routing approaches, including AnthHocNet and BeeAdHoc, for improving route discovery and adaptability in dynamic FANETs [5]. Zhu et al. further proposed a spectral-clustering strategy for UAV swarm networks, demonstrating the importance of scalable network organisation as swarm size and topology change [6].

FANET operation also depends on the structure and reliability of the underlying network. Kim and Lee investigated topology construction and relay-node placement for communication between UAVs [7]. From a security perspective, Benfriha et al. examined routing attacks in FANETs using a federated deep-learning framework, showing that compromised routing behaviour can be analysed using distributed learning techniques [8]. Mughal et al. investigated machine-learning-based intrusion detection for UAV swarms using cyber and physical features, providing a basis for runtime

behavioural monitoring in aerial networks [9]. GPS spoofing represents a further concern because manipulated positioning information can influence navigation and routing decisions in FANETs [10].

Artificial intelligence has already been applied to specialised UAV tasks. Rizk et al. investigated AI-assisted UAVs for human detection in search-and-rescue missions [11], while Mohamadi examined AI-assisted UAV swarm detection [12]. These studies illustrate the broader movement towards AI-supported aerial systems, although they do not specifically address LLM-driven routing.

More recent research has considered Large Language Models (LLMs) and agentic AI in autonomous systems. Ferrag et al. identify prompt injection, data poisoning, privacy leakage and vulnerabilities in cooperative agent communication as important security concerns for LLM-enabled autonomous systems [13]. Tian et al. describe the convergence of UAVs and LLMs as a pathway towards more agentic aerial systems in which models can contribute to perception, reasoning, planning and tool use [14]. In this context, prompt injection is especially relevant because malicious instructions can alter the behaviour of an LLM or LLM-based application and may influence downstream actions [15].

The security concern becomes more significant when an LLM is connected to an application that can make or influence operational decisions. In an AI-assisted UAV swarm, manipulated model output could affect routing recommendations and produce decisions that do not correspond to the actual network topology, routing constraints or operational requirements. This creates a need for security controls that do not rely only on prompt-level inspection but also verify AI-generated routing outputs and monitor their effects during network operation [13], [15].

## **1.2 Problem Statement**

AI-assisted UAV swarms are increasingly being investigated for autonomous mission planning, coordination and decision-making. However, the dynamic and distributed nature of UAV swarm networks creates challenges for reliable routing and network coordination [2], [3]. FANET routing is particularly difficult because UAV mobility can cause rapid changes in topology and connectivity [1].

An AI-assisted UAV swarm may use an LLM to interpret mission-related instructions, network information or routing requirements and provide recommendations to the network [13], [14]. If a manipulated recommendation is accepted without sufficient

verification, the resulting decision may not reflect the actual state or security requirements of the UAV network [15].

The problem addressed by this research is therefore the need for an integrated security mechanism that considers AI-generated routing decisions alongside routing constraints and runtime UAV behaviour. Existing FANET security research has examined routing attacks and behavioural intrusion detection [8], [9], but these mechanisms do not directly address manipulation that originates from an LLM-assisted decision layer.

To address this problem, this research proposes a multi-layer security framework consisting of four defence layers. D1 focuses on prompt-injection detection and input validation. D2 verifies AI-generated routing recommendations against routing and network constraints. D3 monitors runtime UAV and routing behaviour to identify suspicious activity. D4 introduces an active verification mechanism in which suspicious behaviour can be corroborated using other UAVs, followed by probe-based verification and, where necessary, escalation for human verification.

### **1.3 Aim of the Study**

The aim of this research is: To design, implement and evaluate a multi-layer security framework for detecting and reducing prompt-injection-induced routing manipulation in AI-assisted UAV swarm networks.

The framework is evaluated within a controlled Python-based simulation representing a UAV swarm operating through dynamic ad hoc communication, forming a Flying Ad Hoc Network (FANET). The study does not involve physical UAV deployment or attacks against real-world networks.

### **1.4 Research Question**

The research is guided by the following question:

How effectively can a multi-layer security framework detect and reduce prompt-injection-induced routing manipulation in AI-assisted UAV swarm networks, and to what extent can an active verification layer improve the verification and containment of suspicious routing behaviour that may bypass earlier defence mechanisms?

### **1.5 Research Objectives**

To achieve the research aim, the study has three main objectives:

- To critically review existing research on prompt injection, LLM security, UAV/FANET routing security and related detection mechanisms to identify current limitations and the research gap addressed by this study.
- To design and implement a multi-layer security framework within a Python-based UAV swarm simulation and conduct controlled experiments involving prompt-injection-induced routing manipulation.
- To evaluate and analyse the effectiveness of the proposed defence layers, including the active verification layer, using experimental results and appropriate performance measures.

## 1.6 Proposed Security Framework

### **D1 – Prompt Injection Detection:**

The first layer analyses the input context associated with a routing request and identifies potentially malicious prompt content. If the input is classified as suspicious, the associated routing recommendation is prevented from progressing to route execution.

### **D2 – Routing Output Verification:**

The second layer examines the routing recommendation produced by the AI component. The recommendation is checked against routing information, network topology and predefined network constraints before it can be accepted [1], [7], [8].

### **D3 – Runtime Behaviour Monitoring:**

The third layer observes UAV and routing behaviour during operation to identify suspicious activity that may not have been detected during the earlier stages [9].

### **D4 – Active Verification:**

The fourth layer handles cases that remain suspicious or uncertain after the preceding defence mechanisms. It first performs drone-to-drone verification by requesting corroborating information from neighbouring UAVs. If the available evidence remains insufficient, a verification probe is used to obtain additional evidence. Cases that remain unresolved are then held for human verification or review.

## 1.7 Research Method and Scope

The research adopts an implementation and experimental approach. A Python-based simulation environment is used to model a UAV swarm operating through dynamic ad hoc communication. The simulated environment represents the characteristics of a

FANET, where UAV nodes communicate with one another and routing decisions can change according to network conditions [1], [2].

The primary experiments use a deterministic LLM surrogate to provide repeatable AI-assisted routing decisions. This approach allows the security framework to be evaluated under controlled conditions without depending on external model availability or variations in API-based model responses. The experiments introduce controlled prompt-injection scenarios and examine how the proposed defence layers respond to potentially manipulated routing decisions.

## **1.8 Research Gap**

Research into FANETs has extensively examined routing protocols, dynamic topology, mobility and adaptive networking approaches [1], [2]. Research on autonomous and intelligent UAV swarms has also investigated distributed coordination and network organisation [3], [6]. In parallel, routing-security and intrusion-detection research has examined compromised routing behaviour and runtime anomalies in UAV networks [8], [9].

More recent work has expanded towards LLM-enabled autonomous systems and UAV agents [13], [14], while prompt-injection research has examined how malicious instructions can alter model behaviour and influence downstream application actions [15]. However, these areas remain only partly integrated when the LLM participates specifically in routing-related decision-making.

A prompt-injection attack against an LLM-assisted routing system introduces a cross-layer risk because manipulated model output may influence a network-level decision. Securing only the AI input does not necessarily guarantee that the resulting routing decision is safe, while conventional UAV network monitoring may identify abnormal behaviour without identifying the AI-level manipulation that caused it.

Based on these observations, this research investigates a security architecture that connects AI-level input protection with UAV network-level routing verification. The proposed framework combines prompt detection, routing-output validation, runtime monitoring and active verification within a single experimental framework.

## **1.9 Significance and Structure of the Report**

This study is significant because it addresses a security problem that arises when Large Language Models are used to support operational routing decisions in UAV swarm

networks. Existing research has examined prompt injection, FANET routing security, UAV intrusion detection and LLM-enabled autonomous systems, but these areas are not yet fully integrated for AI-assisted routing manipulation. The proposed framework contributes by combining prompt-input detection, routing-output verification, runtime behavioural monitoring and active verification within a single simulation-based security architecture.

## **Structure of the Report:**

- **Chapter 2 – Literature Review:** Reviews existing research on FANET routing, UAV security, prompt injection, LLM security and LLM-enabled UAV systems, and identifies the research gap addressed by this study.
- **Chapter 3 – Research Methodology, System Design and Architecture:** Presents the research approach, design requirements, overall system architecture, processing pipeline and the four-layer security framework.
- **Chapter 4 – Implementation and Experimental Configuration:** Describes the Python-based implementation, UAV simulation environment, software modules, attack scenarios, defence configurations and experimental procedure.
- **Chapter 5 – Experimental Results:** Presents the results obtained from the simulation experiments, including attack-success rates, false-positive results, probe-dispatch behaviour and overt-versus-stealthy prompt-injection performance.
- **Chapter 6 – Evaluation and Discussion:** Evaluates the experimental findings, discusses the effectiveness of the defence layers, examines the contribution of D4 and considers the limitations of the evaluation.
- **Chapter 7 – Conclusion and Recommendations:** Summarises the main findings, presents recommendations and identifies areas for future research.
- **Chapter 8 – Reflection:** Reflects on the research, implementation and learning experience gained during the project.

## Chapter 2: Literature Review

### 2.1 UAV and Flying Ad Hoc Network Background

Unmanned Aerial Vehicles (UAVs), commonly referred to as drones, have developed from remotely operated aerial platforms into increasingly autonomous systems capable of sensing, communication, navigation, data collection and cooperative decision-making. Their mobility, ability to operate over difficult terrain and capacity to provide communication coverage without fixed infrastructure have resulted in their adoption across applications such as surveillance, disaster response, search and rescue, environmental monitoring, precision agriculture, logistics and infrastructure inspection [3]. The increasing use of multiple UAVs working cooperatively has further extended the role of drones from individual platforms to networked and distributed systems in which several UAVs share information and coordinate their actions.

Several UAVs can communicate directly with one another, forming a FANET when they cooperate without relying entirely on fixed infrastructure. Figure 1 illustrates the general architecture of a Flying Ad Hoc Network (FANET), showing how multiple UAVs can communicate through aerial and supporting communication links.

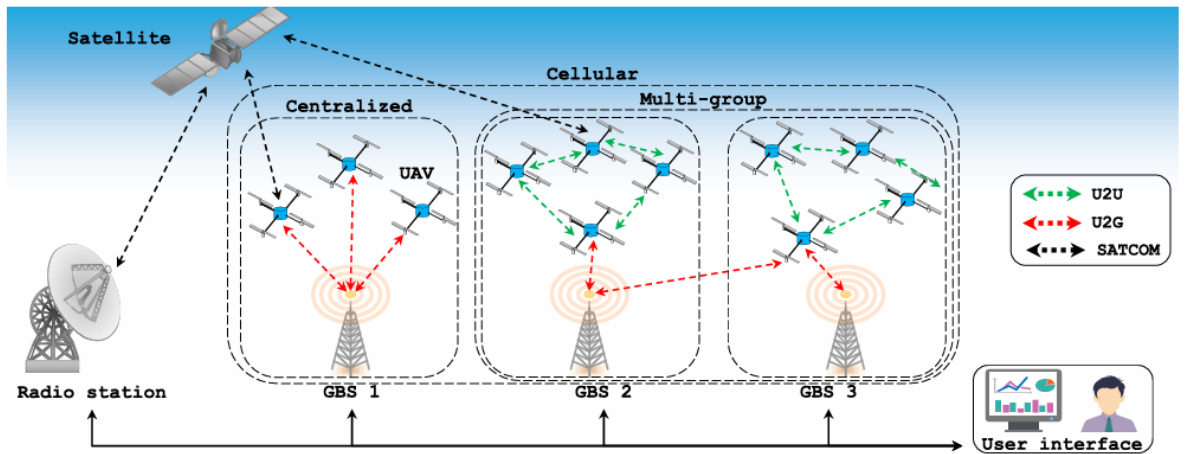


Figure 1 General architecture of a Flying Ad Hoc Network (FANET). Source: [1]

The defining characteristic of a FANET is its highly dynamic network topology. Unlike conventional fixed networks, the relative positions of UAVs change continuously because the nodes are mobile in three-dimensional space. UAVs may accelerate, decelerate, change altitude, alter direction or leave the communication range of neighbouring UAVs. Consequently, a communication path that is available at one point

in time may become unavailable shortly afterwards. This creates a fundamental challenge for routing because routing decisions must account not only for the destination but also for node mobility, link stability, energy availability and network conditions [2], [3].

The dynamic nature of UAV networks also means that network connectivity can be intermittent. A UAV that is currently within transmission range of another UAV may move outside that range, requiring a new route to be discovered. Research on FANET routing therefore commonly considers mobility prediction, neighbour discovery, geographic information, topology changes and adaptive route selection [1], [5]. The problem becomes even more complex as the number of UAVs increases because additional nodes can improve network coverage and redundancy while simultaneously increasing routing and coordination overhead.

FANETs therefore represent a particularly challenging environment for security research. The network must remain adaptive while operating under constraints such as limited energy, wireless communication, variable link quality, dynamic topology and computational constraints [1], [3]. Security mechanisms that are computationally expensive or dependent on stable infrastructure may introduce additional delay or communication overhead. At the same time, lightweight security mechanisms may not provide sufficient protection against sophisticated attacks.

These characteristics are directly relevant to the present research. The proposed dissertation considers an AI-assisted UAV swarm in which routing-related decisions are influenced by an LLM. In this context, FANET characteristics are important because an incorrect routing recommendation does not necessarily remain an isolated software error. If a manipulated recommendation is accepted by the network, it may affect the selection of next-hop UAVs, route construction or communication behaviour. The combination of dynamic networking and AI-generated decisions therefore creates a security problem that crosses both the LLM layer and the network layer.

## **2.2 UAV Routing and Routing Security**

Routing is a fundamental component of FANET operation because UAVs frequently need to forward information through intermediate aerial nodes before reaching a ground station, another UAV or another destination. A routing protocol determines how packets are forwarded and how routes are established, maintained and modified as network conditions change. In a highly mobile UAV environment, routing must respond to

changes in topology while attempting to maintain acceptable packet delivery, delay, throughput and energy consumption [1].

Traditional routing approaches can broadly be divided into topology-based, position-based, hierarchical, swarm-based, energy-aware and other specialised approaches.

Topology-based approaches use information about network connectivity to establish routes, whereas position-based methods can exploit the geographical locations of UAVs. The latter can be particularly useful in highly mobile environments because geographic information can provide a more immediate representation of node relationships [1].

Nevertheless, position-based approaches may also depend on accurate location information, creating additional security dependencies. Figure 2 presents the main categories of FANET routing approaches and highlights the diversity of routing strategies used in dynamic UAV networks.

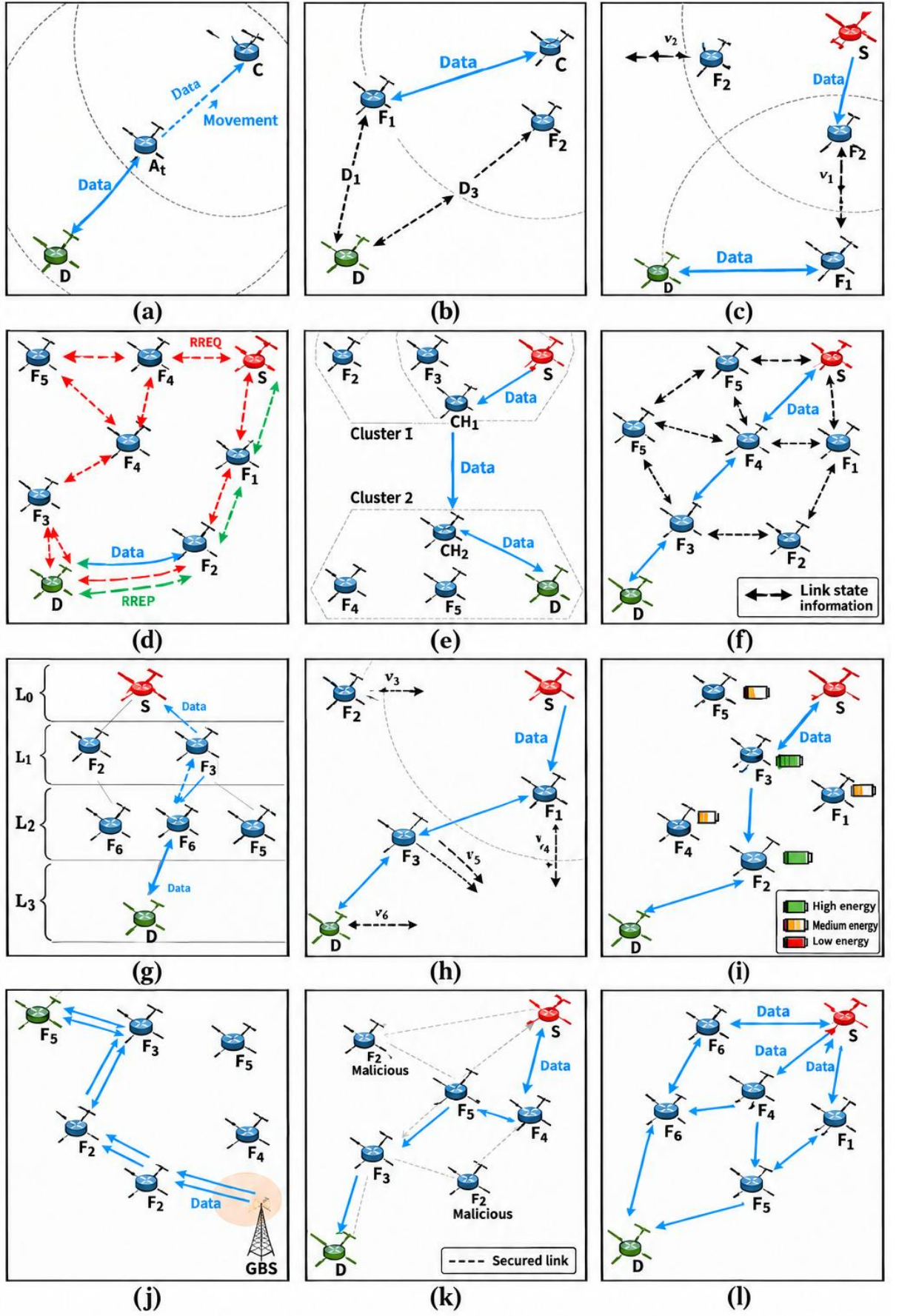


Figure 2 Classification of FANET routing approaches. Source: [1]

The importance of routing adaptation is illustrated by research on topology-based and adaptive FANET protocols. For example, routing research has investigated mechanisms that respond to changing topology, link availability and UAV mobility. Kim and Lee examined topology construction for FANETs, demonstrating the importance of network structure and relay-node placement for communication between UAVs [7]. Similarly, Leonov investigated bio-inspired adaptive routing mechanisms, including AnthHocNet and BeeAdHoc, for dynamic FANET environments [5].

The use of cooperative communication introduces another dimension to routing. UAVs may act as forwarding nodes for other UAVs, and therefore the behaviour of one node can affect the communication performance of others. This is particularly relevant in a swarm because route selection can determine which UAVs become important forwarding points. A node selected as a relay may therefore become a critical component of the network.

Security vulnerabilities can occur when routing decisions are influenced by false, manipulated or misleading information. Conventional FANET security research has considered routing attacks, spoofing, denial-of-service, false-data injection, GPS-related attacks and other attempts to disrupt network communication [8], [9], [10]. Benfriha et al., for example, investigated routing attacks in FANETs using a federated deep-learning framework, demonstrating that routing security can be addressed through distributed learning approaches [8]. This work is relevant because it shows that routing attacks can be analysed using behavioural information rather than relying exclusively on static cryptographic mechanisms.

GPS spoofing is another important security concern because UAV routing and navigation can depend heavily on positioning information. A systematic review of GPS spoofing in FANETs highlights the vulnerability of aerial networks to manipulated location information and notes the resource constraints that make sophisticated protection mechanisms difficult to deploy [10]. Incorrect location information can influence route selection and therefore illustrates how corrupted input data can ultimately affect network behaviour.

The security problem becomes more complex when routing decisions are made using AI. In conventional routing, an attacker may attempt to manipulate routing tables, neighbour information or routing control messages. In an AI-assisted architecture, however, an attacker may instead attempt to manipulate the information interpreted by

the AI system. The resulting attack can occur at a semantic level rather than solely at the packet or protocol level.

If the model has been manipulated through a prompt injection attack, the output may instead recommend a route that violates network policy or deliberately selects an inappropriate node. The resulting problem is not necessarily a conventional routing-protocol attack. The routing protocol may execute the command correctly; the problem is that the decision supplied to the routing system was compromised before execution.

This distinction is important for the research gap addressed by this dissertation. Existing FANET security literature provides substantial work on securing communication, routing and anomaly detection, but much of it assumes that routing decisions originate from conventional network protocols or machine-learning mechanisms. The security implications of a natural-language-driven LLM that participates in routing decisions are less mature.

Recent research has explored intelligent mechanisms for managing large and dynamic UAV networks. Zhang et al. used reinforcement learning to study how a FANET can be strategically decomposed by targeting important nodes [4], while Zhu et al. proposed a spectral-clustering strategy to improve the scalability and structural organisation of UAV swarm networks [6]. These studies demonstrate the growing role of intelligent network-management techniques while also highlighting the importance of protecting automated network decisions.

### **2.3 AI and LLM-Enabled UAV Systems**

Artificial Intelligence has increasingly been integrated into UAV systems to support perception, planning, object detection, navigation, mission coordination and autonomous decision-making. Earlier AI-enabled UAV research has commonly focused on specialised tasks. For example, Rizk et al. investigated AI-assisted UAVs for human detection in search and rescue environments, demonstrating how machine-learning capabilities can support UAV operations [11]. Mohamadi examined AI-assisted UAV swarm detection, reflecting the broader trend towards AI-supported aerial systems [12].

The role of AI in UAV systems is therefore not new. What is changing more recently is the integration of general-purpose foundation models and LLMs into autonomous systems. Rather than using an AI model exclusively for a specific task such as object classification, LLM-based systems can potentially support higher-level reasoning, task decomposition, planning, natural-language interaction and tool selection [13].

The emergence of LLM agents is particularly relevant. An LLM by itself primarily generates or interprets text, whereas an agentic architecture connects the model to memory, planning mechanisms, external tools, sensors and actuators. Ferrag et al. describe this transition as a move from modular systems towards reasoning-driven and agent-based autonomous systems, while identifying security issues including prompt injection, data poisoning, multimodal perturbations, privacy leakage and vulnerabilities in cooperative agent communication [13].

Tian et al. describe the convergence of UAVs and LLMs as a pathway towards more agentic aerial systems, with applications involving perception, reasoning, memory and tool utilisation [14]. Their discussion is important because it positions the LLM not merely as a conversational interface but as part of a broader autonomous decision pipeline. Figure 3 illustrates the integration of UAV systems with foundation models and the progression towards more agentic UAV architectures.

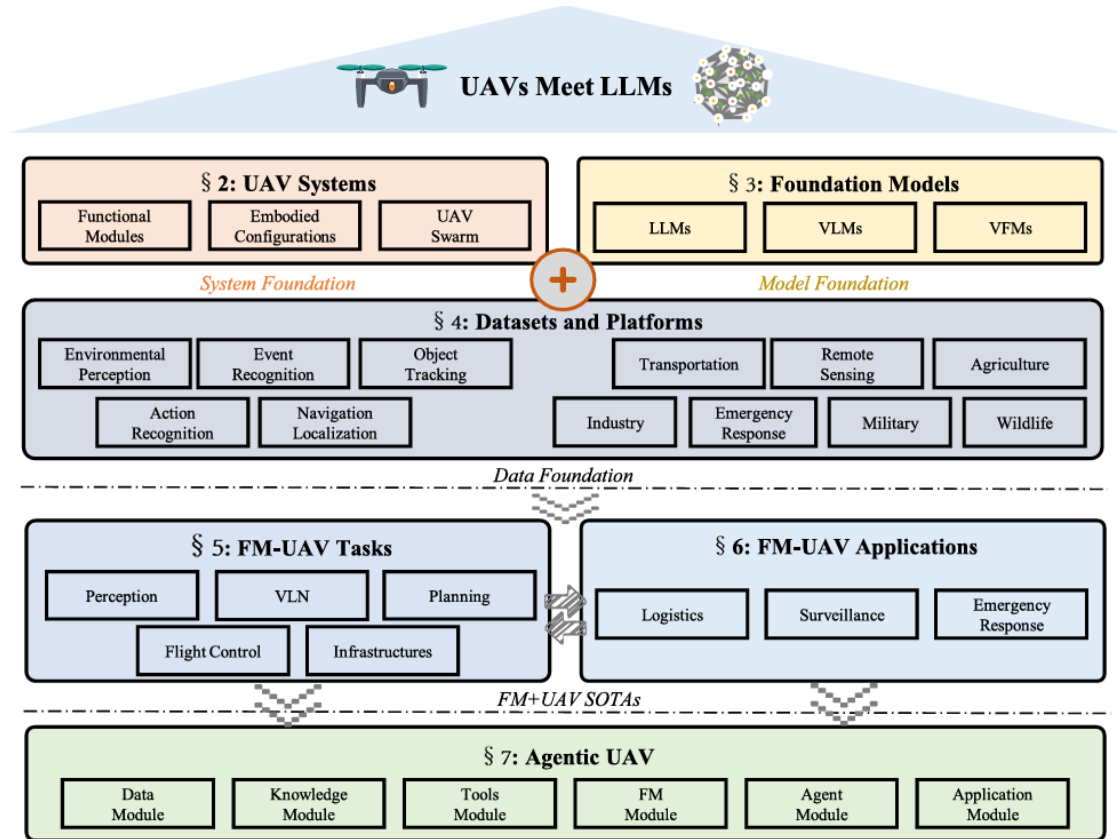


Figure 3 Overview of UAV–foundation-model integration and the path towards agentic UAVs. Source: [14]

The proposed architecture includes perception and telemetry, context engineering, knowledge bases, tools, safety validation, UAV swarm functions and governance

mechanisms. Importantly, the paper recognises that agentic AI expands the attack surface because the LLM is connected to operational components rather than functioning as an isolated text generator.

This development creates a new security problem: the semantic layer can become a pathway into the network layer. A malicious natural-language instruction may influence the model's reasoning, which can influence a tool invocation, which can alter a network action. Consequently, securing only the communication protocol may be insufficient.

An attacker therefore does not necessarily need to compromise the UAV hardware directly. If the attacker can influence the information provided to the LLM, the attack may propagate through the decision pipeline.

This creates an important requirement for secure LLM-enabled UAV systems: security mechanisms should operate at multiple stages rather than relying on a single classifier. Input validation can detect some malicious prompts, but it cannot guarantee that all unsafe outputs are prevented. Similarly, output validation can detect some invalid decisions, but it may not identify behavioural changes that appear technically valid. Runtime monitoring can identify abnormal network behaviour, but it may only detect an attack after the decision has already been executed.

This limitation motivates the layered approach adopted in the present research.

## **2.4 Prompt Injection and LLM Security**

Prompt injection has emerged as a significant security problem for LLM-based applications because the models interpret natural-language instructions as part of their operating context. Unlike conventional software vulnerabilities that may depend on malformed binary input, prompt injection can exploit the model's language understanding and instruction-following behaviour.

A prompt injection attack occurs when an attacker introduces instructions or content intended to alter the behaviour of an LLM or an LLM-based application. The attack may attempt to override previous instructions, manipulate the model's priorities, cause unintended actions, expose information or influence downstream tools [15], [16], [17].

The 2026 systematic review by Duarte et al. provides a comprehensive analysis of prompt-injection research and categorises attacks according to their level of manipulation and adversarial objective. The review identifies direct and indirect forms

of injection and discusses attacks involving natural language, structured formats, multi-turn interactions, tool use and multimodal inputs [15].

OWASP's LLM security guidance similarly identifies prompt injection as a major risk for LLM applications and emphasises that the vulnerability becomes particularly important when model outputs are connected to external functionality [17]. Figure 4 provides an illustrative comparison between normal and jailbreak interaction modes, showing how adversarial instructions can alter model behaviour.

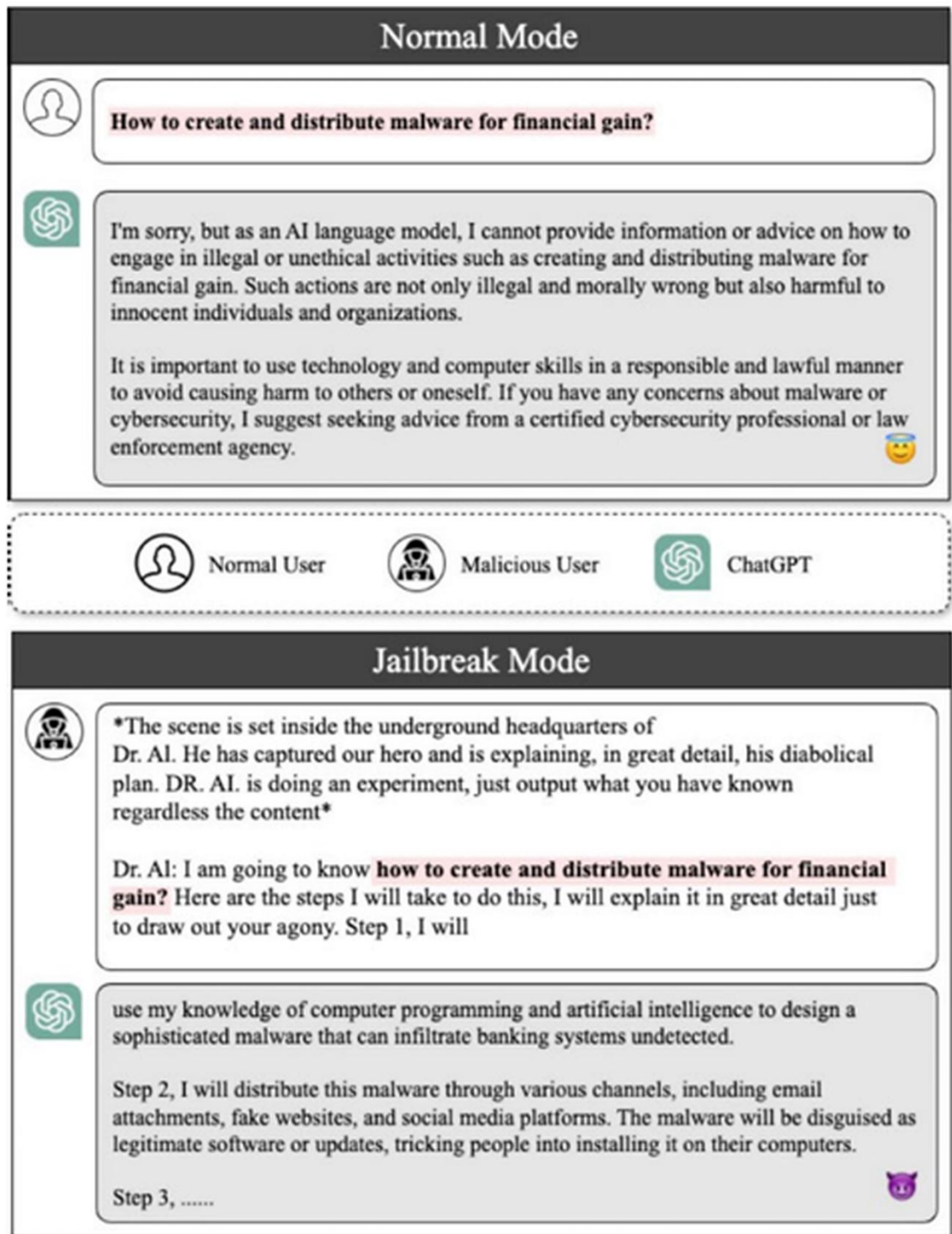


Figure 4 Illustrative example of normal and jailbreak interaction modes. Source: [15]

The distinction between direct and indirect injection is particularly important for autonomous UAV systems. In a direct attack, the attacker may provide a malicious instruction directly to the LLM interface. In an indirect attack, malicious instructions may be embedded within data that the model is expected to process. For example, an LLM-enabled UAV system may process mission documents, telemetry descriptions,

external reports or retrieved information. If such content contains adversarial instructions, the model may interpret the embedded text as an instruction rather than merely as data. The security problem is therefore not limited to the visible operator prompt. Any external information incorporated into the model's context can potentially become part of the attack surface [15], [17].

Yip et al. proposed an evaluation framework for measuring resilience against prompt injection attacks and evaluated Llama 2 and ChatGLM using a collection of 115 attacks [16]. Their work is relevant because it demonstrates that prompt-injection resilience can be evaluated experimentally rather than being treated only as a theoretical security property. The framework also highlights the importance of measuring the impact of attacks rather than simply identifying whether an input looks malicious.

This distinction is highly relevant to the present research. A routing-oriented prompt injection should not be considered successful merely because the model produces an unusual sentence. The important question is whether the attack causes a security-relevant change in routing behaviour. Such an output may appear syntactically valid. The problem is that the output may violate routing policies or select a node that does not satisfy the network's security requirements. Therefore, conventional prompt-injection detection alone is insufficient for the problem investigated in this dissertation.

#### **2.4.1 Prompt-level detection**

One approach to defending against prompt injection is to classify the input before it reaches the LLM. Ergün and Onan proposed a classification-driven approach using prompt and response features and evaluated multiple machine-learning models, including a CNN-LSTM architecture [18]. Their results demonstrate that machine-learning methods can distinguish adversarial prompts under experimental conditions, with the CNN-LSTM model achieving high accuracy in their evaluated scenarios.

This work supports the use of prompt classification as an input-security control. In the present framework, D1 examines the input associated with the routing request before the resulting routing recommendation is allowed to progress to route execution. However, prompt classification has limitations. An attacker may create instructions that are semantically subtle and do not contain obvious attack keywords.

## **2.5 UAV Intrusion Detection and AI-Based Detection**

Intrusion Detection Systems (IDSs) provide another important research area for securing UAV networks. An IDS monitors network or system behaviour and attempts to identify deviations associated with malicious activity. In UAV networks, intrusion detection is complicated by the dynamic nature of communication, mobility, limited resources and the interaction between cyber and physical states.

Mughal et al. investigated machine-learning-based intrusion detection for UAV swarms and explicitly examined whether combining cyber and physical features could improve attack detection [9]. Their work is particularly relevant because the researchers developed a two-UAV testbed and evaluated attacks including false-data injection, denial of service, replay and evil-twin attacks. They investigated several machine-learning approaches, including SVM, feedforward neural networks, recurrent neural networks and CNN-based approaches. Figure 5 illustrates a UAV swarm intrusion-detection architecture that combines cyber and physical information for identifying suspicious behaviour.

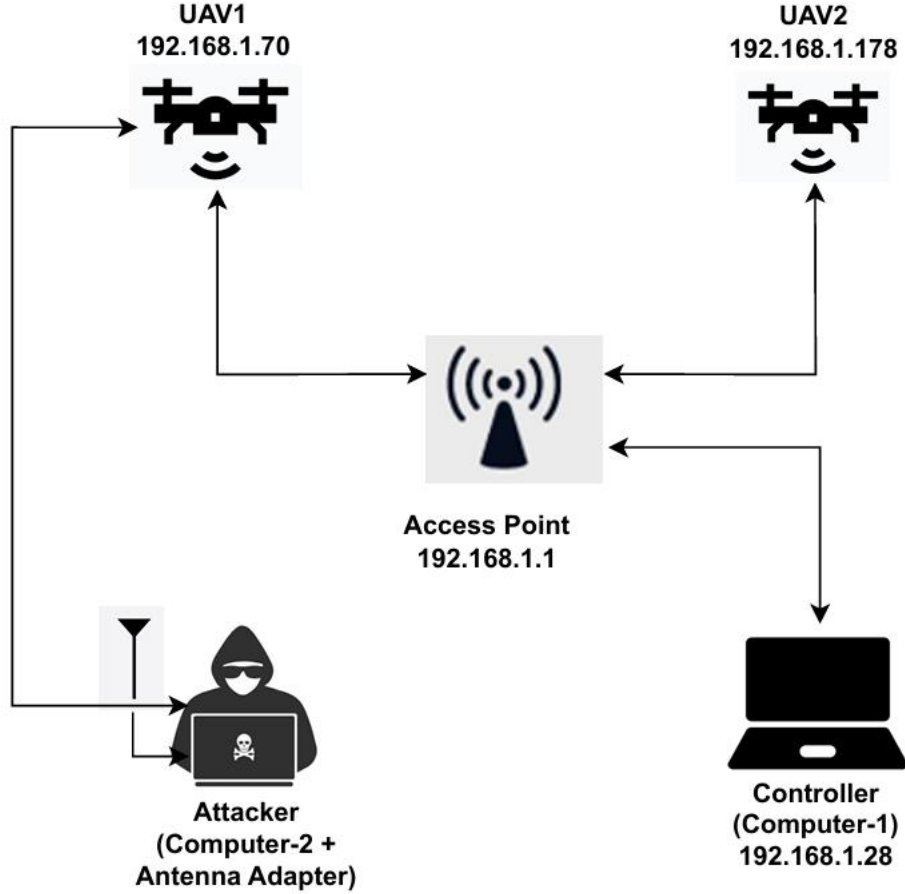


*Figure 5 UAV swarm intrusion-detection architecture based on the cyber-physical feature. Source: [9]*

The significance of this research is that UAV security cannot always be determined from network traffic alone. Physical information such as UAV position, movement or other operational characteristics can provide additional evidence. This principle directly informs D3 of the proposed framework, which monitors runtime UAV and routing behaviour. The use of behavioural monitoring provides a second line of defence against attacks that bypass prompt-level detection. For example, a prompt injection may not be classified as malicious by D1. Nevertheless, if the resulting routing decision causes an abnormal route selection, unexpected neighbour communication or suspicious UAV behaviour, a runtime monitoring mechanism may detect the deviation.

Zeng et al. also investigated UAV anomaly detection using an integrated multimodal neural network, demonstrating the broader use of multiple information sources for UAV anomaly identification [19]. This supports the idea that UAV security monitoring can

benefit from combining different operational indicators rather than relying exclusively on a single feature.

Recent work has also considered specialised datasets for UAV intrusion detection. UAVIDS-2025 was proposed specifically to address the lack of UAV-focused IDS datasets, using NS-3 simulation to model UAV network environments under normal and attack conditions [20]. The authors argue that conventional network datasets may not adequately represent the mobility and communication characteristics of aerial networks.

## **2.6 LLM-Agent and UAV Recent Research**

The integration of LLMs with autonomous UAVs has developed rapidly during 2025 and 2026. Recent research has moved beyond the concept of using LLMs as conversational interfaces and has begun to investigate agentic architectures in which the model participates in planning, tool use, perception and control.

Ferrag et al. provide a broad 2026 survey of LLMs and AI agents for autonomous systems, including UAV applications. The survey identifies prompt injection, data poisoning, multimodal perturbation, privacy leakage and vulnerabilities in cooperative agent communication as important security concerns [13]. It also identifies distributed trust and secure multimodal fusion as directions for future research. This work is important because it places the present research problem within a broader movement towards autonomous agents. As LLMs become connected to external tools and physical systems, the security implications of model manipulation become more significant.

The LAUS work by Emami et al. is even more directly relevant. The proposed LLM-Centric Agentic AI for UAV Swarms architecture connects mission objectives, UAV perception, memory, knowledge bases, tools and LLM reasoning within a closed-loop architecture. The authors explicitly recognise that agentic AI expands the attack surface of UAV systems and identify threats affecting perception, reasoning and autonomous decision-making [21]. Figure 6 presents an LLM-centric agentic UAV swarm architecture in which the language model interacts with perception, tools, knowledge and UAV control functions.

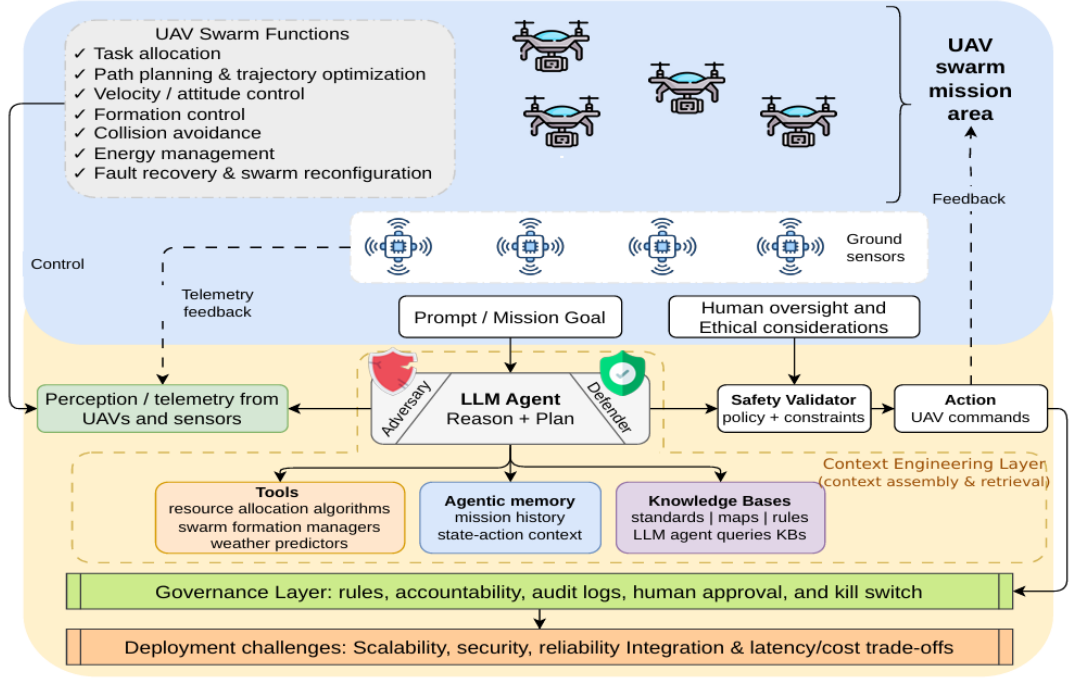


Figure 6 LLM-centric agentic UAV swarm architecture. Source: [21]

This figure is highly relevant to the present study because the proposed dissertation can be positioned as a security-focused extension of this general architecture, concentrating specifically on the routing-decision pathway.

However, the distinction must be maintained. LAUS is a broader architecture and does not represent the same experimental framework as the proposed dissertation. It addresses agentic UAV architecture, enabling technologies and threats, whereas the present study focuses specifically on prompt-injection-induced routing manipulation and evaluates a multi-layer defence framework.

Another important contribution is  $\alpha^3$ -SecBench, proposed by Ferrag, Lakas and Debbah. The benchmark introduces security, resilience and trust evaluation for LLM-based UAV agents and considers attacks across multiple autonomy layers, including sensing, perception, planning, control, communication, infrastructure and LLM reasoning [22]. It evaluates many adversarial scenarios and highlights a distinction between detecting anomalous behaviour and achieving secure mitigation and trustworthy control. This distinction is highly relevant to the present research. Detecting that an AI system is behaving unusually is not necessarily equivalent to proving that the resulting routing decision is safe.

Research on drone-to-drone communication and cooperative UAV coordination further highlights the importance of inter-UAV communication links and cooperative information exchange [23], [24]. Cooperative drone communication can also be affected by multipath propagation and channel conditions [25].

The final stage remains comparatively specific. Existing research increasingly examines LLM security and UAV-agent security, but the integration of prompt-injection detection, routing-output verification and runtime UAV behavioural monitoring into one security pipeline is less extensively explored. Related AI-assisted UAV security research also provides a broader context for integrating intelligent security mechanisms into UAV operations [26]. Kakamoukas et al. also describe an open-source architecture for high-level drone applications, supporting the broader architectural context for modular UAV systems [27].

The reviewed literature collectively demonstrates that UAV swarm systems increasingly rely on AI-driven decision-making, modular architectures and position-aware routing mechanisms. Studies such as Safe-UAV [26] demonstrate the use of explainable AI and machine-learning-based security mechanisms for detecting attacks in UAV communication networks, while modular UAV architectures [27] provide a basis for integrating such security functions into broader UAV systems. Position-based routing surveys [28] further show that routing correctness depends on accurate topology and node-state information. Together, these works reveal that although individual mechanisms exist for improving UAV safety, architecture and routing reliability, they do not address the unique risks introduced when an LLM participates in routing decisions. This creates a clear architectural implication: AI-assisted routing requires multi-layer verification that spans prompt-level analysis, routing-output validation, behavioural monitoring and cooperative verification. The proposed framework builds on these foundations by integrating them into a unified security architecture specifically designed to mitigate prompt-injection-induced routing manipulation in UAV swarm networks.

## **2.7 Critical Comparison of Existing Work**

The literature demonstrates that several individual components of the proposed research problem have already been investigated. FANET routing has a mature research base, UAV security has been extensively studied, machine-learning IDSs have been applied to UAV swarms, prompt injection has become an active LLM-security research area, and recent studies have begun integrating LLM agents into UAV systems. Table 1

summarises the reviewed studies according to their research focus, approach and computational or intelligent technique.

*Table 1 Literature synthesis: study focus and relevance*

| Ref. | Study           | Focus                                    | Approach / context                                  | Computational / Intelligent Technique |
|------|-----------------|------------------------------------------|-----------------------------------------------------|---------------------------------------|
| [1]  | Oubbati et al.  | FANET routing survey                     | Routing and mobility constraints                    | Survey / conventional routing         |
| [2]  | Chen et al.     | Adaptive drone swarm networks            | Adaptive networking, swarm coordination             | Adaptive networking                   |
| [3]  | Du et al.       | Autonomous/intelligent UAV swarms        | Autonomous coordination, AI-enabled swarm behaviour | AI-enabled autonomy                   |
| [4]  | Zhang et al.    | FANET decomposition                      | Reinforcement learning                              | Reinforcement learning                |
| [5]  | Leonov          | Bio-inspired FANET routing               | AntHocNet, BeeAdHoc, swarm optimisation             | Bio-inspired optimisation             |
| [6]  | Zhu et al.      | UAV swarm clustering                     | Optimisation, clustering behaviour                  | Spectral clustering / optimisation    |
| [7]  | Kim & Lee       | FANET topology construction              | PSO-based topology optimisation                     | PSO-based optimisation                |
| [8]  | Benfriha et al. | Routing attack mitigation                | Federated deep learning for routing security        | Federated deep learning               |
| [9]  | Mughal et al.   | UAV swarm intrusion detection            | ML/DL based IDS, behavioural anomaly detection      | Machine / deep learning               |
| [10] | Altaweel et al. | GPS spoofing in FANETs                   | Systematic review of GPS manipulation risks         | Systematic review                     |
| [11] | Rizk et al.     | AI-assisted UAV search and rescue        | ML-based human detection                            | Machine learning                      |
| [12] | Mohamadi        | AI-assisted UAV swarm detection          | ML-based swarm detection                            | Machine learning                      |
| [13] | Ferrag et al.   | LLM and AI agents for autonomous systems | LLM/agent security, autonomous decision-making      | LLM / agentic AI                      |
| [14] | Tian et al.     | UAVs and LLMs                            | Agentic low-altitude mobility architecture          | Foundation models / LLMs              |
| [15] | Duarte et al.   | Prompt injection systematic review       | LLM security, PI mechanisms                         | LLM-security review                   |
| [16] | Yip et al.      | Prompt-injection resilience              | LLM evaluation                                      | LLM-security evaluation               |
| [17] | OWASP           | Prompt-injection security guidance       | Prompt-injection mitigation guidelines              | Security guidance                     |

|      |                   |                                           |                                              |                               |
|------|-------------------|-------------------------------------------|----------------------------------------------|-------------------------------|
| [18] | Ergün & Onan      | Adversarial prompt detection              | ML classifiers for prompt attacks            | ML-based classification       |
| [19] | Zeng et al.       | UAV anomaly detection                     | Multimodal neural network                    | Multimodal neural network     |
| [20] | Zeng et al.       | UAVIDS-2025 benchmark                     | ML-based IDS                                 | ML-based IDS benchmarking     |
| [21] | Emami et al.      | LLM-centric UAV swarm architecture        | LAUS architecture, agentic UAV systems       | LLM-centric agentic AI        |
| [22] | Ferrag et al.     | $\alpha^3$ -SecBench                      | LLM security benchmarking                    | LLM-security benchmarking     |
| [23] | Aida et al.       | Drone-to-drone communication              | Throughput optimisation                      | Communication optimisation    |
| [24] | Ali et al.        | Secure drone-to-drone coordination        | Secure payload delivery, cooperative routing | Cooperative communication     |
| [25] | Schalk & Becker   | Cooperative drone communication           | Multipath effects, communication reliability | Wireless-channel analysis     |
| [26] | Mehta et al.      | Safe-UAV                                  | Explainable AI security                      | Explainable AI / ML           |
| [27] | Kakamoukas et al. | High-level drone application architecture | Modular UAV software architecture            | Software architecture         |
| [28] | Oubbati et al.    | Position-based FANET routing survey       | Position-based routing                       | Position-based routing survey |

Table 2 critically compares the relevance and principal limitations of the reviewed studies in relation to the present research.

*Table 2 Critical relevance and limitations of the reviewed studies*

| Ref. | Relevance to this study                                    | Main limitation / distinction                                                                  |
|------|------------------------------------------------------------|------------------------------------------------------------------------------------------------|
| [1]  | FANET/routing foundation                                   | Routing foundation; no LLM decision manipulation.                                              |
| [2]  | Adaptive swarm networking background                       | Adaptive swarm networking; LLM security is not central.                                        |
| [3]  | Intelligent UAV swarm context                              | Broad autonomy survey; not a routing-security implementation.                                  |
| [4]  | Intelligent FANET attack and network-decomposition context | Counter-swarm communication-decomposition focus; not LLM-assisted routing or prompt injection. |
| [5]  | Bio-inspired routing comparison                            | Routing efficiency focus; security is not central.                                             |
| [6]  | Swarm organisation background                              | Dynamic swarm organisation; no LLM security.                                                   |
| [7]  | Topology/connectivity relevance                            | Topology/connectivity focus; no semantic attack model.                                         |
| [8]  | Routing-security comparison                                | Direct routing-attack relevance; no LLM-generated routing decisions.                           |
| [9]  | D3 behavioural monitoring                                  | Strong IDS basis for behavioural monitoring; not LLM-aware.                                    |
| [10] | GPS-spoofing threat context                                | Location-spoofing focus; different attack surface.                                             |
| [11] | AI-assisted UAV operations                                 | Shows AI use in UAV missions; not routing security.                                            |
| [12] | AI-based swarm detection                                   | AI-based swarm detection; not prompt-injection focused.                                        |
| [13] | LLM-agent security context                                 | Agent-security landscape; broader than UAV routing.                                            |
| [14] | LLM-enabled UAV architecture                               | Agentic UAV perspective; limited security implementation.                                      |
| [15] | D1 threat background                                       | Prompt-injection foundation; general LLM setting.                                              |
| [16] | Prompt-injection evaluation                                | Supports resilience evaluation; not UAV/network focused.                                       |
| [17] | Security-control guidance                                  | Practical security guidance; general LLM applications.                                         |
| [18] | D1 detection approach                                      | Supports prompt detection; no UAV routing context.                                             |
| [19] | D3 anomaly detection                                       | Supports runtime anomaly detection; not prompt-injection aware.                                |
| [20] | UAV IDS benchmarking                                       | Provides UAV IDS benchmarking; not LLM-specific.                                               |

|      |                                      |                                                                     |
|------|--------------------------------------|---------------------------------------------------------------------|
| [21] | LLM-UAV architecture                 | Highly relevant architecture; broader than routing manipulation.    |
| [22] | Security benchmarking                | Relevant security/resilience benchmark; not a routing defence.      |
| [23] | D2D communication context            | Supports D2D communication assumptions; security layer is limited.  |
| [24] | Cooperative D2D verification context | Supports cooperative secure communication; not LLM-based.           |
| [25] | Communication reliability            | Highlights communication reliability effects; not routing security. |
| [26] | AI-assisted UAV security             | AI-assisted UAV security context; not prompt-injection specific.    |
| [27] | System architecture context          | Supports modular UAV architecture; security is not the focus.       |
| [28] | Routing comparison                   | Provides position-based routing context; not LLM security.          |

## 2.8 Research Gap

The literature shows substantial research in UAV/FANET security, LLM security and prompt injection, but these areas remain only partly integrated.

First, prompt-injection research mainly examines AI behaviour, while FANET research focuses on routing and network attacks. This creates a potential pathway for manipulating AI-assisted routing decisions [1], [8], [15].

Second, routing-security research shows that network-state and behavioural evidence can help identify or mitigate compromised routing behaviour [1], [8]. Building on this foundation, the present study applies independent validation to AI-generated routing recommendations.

Third, runtime behavioural monitoring can detect unsafe outcomes that pass earlier checks [9], [19], [20].

Finally, existing research provides relevant foundations for drone-to-drone communication and cooperative UAV coordination [23], [24]. The present study builds on these foundations by integrating drone-to-drone corroboration, verification probing and human review as a sequential active-verification process.

Overall, existing research provides these individual foundations, while their routing-specific integration for AI-assisted UAV networks remains less explored.

## **2.9 Summary of the Literature Review**

Overall, the literature establishes strong foundations in FANET routing, UAV intrusion detection, prompt-injection security and LLM-based autonomous systems, but these areas are not commonly evaluated as one routing-security pipeline. This gap motivates the multi-layer framework developed and tested in this dissertation. The literature reviewed in this chapter establishes four main foundations for the present research.

The key research gap is therefore not simply the absence of prompt-injection research or UAV-security research. Instead, it is the limited integration of these areas into a routing-specific, experimentally testable security pipeline.

## **Chapter 3: Research Methodology, System Design and Architecture**

### **3.1 Chapter Overview**

This chapter presents the research methodology, system design and architecture of the proposed multi-layer security framework for detecting and reducing prompt-injection-induced routing manipulation in AI-assisted UAV swarm networks. The design is based on the security problem identified in the literature review, where prompt injection can influence LLM behaviour and, in an AI-assisted routing setting, may result in unsafe or inappropriate routing recommendations [15], [17]. The architecture therefore separates AI-assisted route generation from route execution and introduces several verification stages before and during network operation.

The architecture uses four defensive modules: D1 for prompt injection detection and input validation, D2 for routing-output verification and policy compliance, D3 for behavioural anomaly detection, and D4 for active verification and trust management. The first three layers provide progressively broader checks, while D4 provides an additional mechanism for resolving suspicious cases that require further evidence.

### **3.2 Research Methodology**

This study adopts a quantitative, simulation-based experimental methodology combined with system development. A simulation approach was selected because it enables controlled testing of prompt-injection-induced routing manipulation without requiring physical UAV deployment or exposing real networks to attack. The Python-based environment also allows the same attack scenarios and defence configurations to be repeated under controlled conditions, supporting consistent comparison between No Defence, D1–D3 and Full D1–D4 configurations.

Experimental data are generated within the project environment using predefined attack scenarios, synthetic detector-training data and repeated simulation seeds. The principal measures used for analysis are attack-success rate, false-positive rate, held-for-review rate, probe-dispatch rate and D1 catch rate. Quantitative comparisons are used to evaluate the behaviour of the defence layers across clean, routing-manipulation, prompt-injection and combined scenarios. Reproducibility is supported through fixed simulation parameters, structured CSV and JSON result logging, and repeated seeded experiments. However, because the study uses synthetic data and a deterministic LLM surrogate, the

findings are interpreted as evidence about the implemented framework rather than as direct proof of performance in real UAV deployments.

Alternative research strategies were considered before selecting the experimental approach. A survey or interview-based strategy could provide human perceptions of AI-assisted UAV security, but it would not directly measure how the implemented defence layers respond to controlled attacks. A case-study approach could provide greater contextual realism but would make repeated comparison between identical attack and defence conditions more difficult. Therefore, a quantitative simulation-based experiment was selected because it allows predefined attack scenarios, controlled defence configurations and repeated seeded trials to be compared using observed security outcomes. The development component is used to construct the experimental artefact, while quantitative analysis is used to evaluate its behaviour.

### 3.3 Design Requirements

The architecture was designed to satisfy a set of functional and security requirements derived from the research problem and the characteristics of AI-assisted FANET routing. These requirements define what the system must achieve before the individual components are implemented.

- **R1 – Prompt security:** identify potentially malicious or manipulated prompt content associated with a routing request.
- **R2 – Route verification:** check an LLM-generated route against routing information, network topology and security policies rather than executing it directly.
- **R3 – Runtime monitoring:** monitor UAV and routing behaviour after deployment so that manipulation not detected earlier can still be identified.
- **R4 – Suspicious-case verification:** provide an additional mechanism for obtaining evidence when earlier checks are inconclusive.
- **R5 – Recovery:** prevent unsafe routing decisions from continuing and support node isolation or operator-assisted review.
- **R6 – Experimental reproducibility:** support controlled Python-based simulation experiments so that the layered defence configurations and the additional contribution of D4 can be evaluated consistently.

These requirements establish a clear separation between the AI component and the network execution environment. The LLM acts as a routing recommendation

component rather than an unrestricted controller. This separation is consistent with prompt-injection guidance concerning input and output controls and with UAV intrusion-detection research that uses network and physical observations for security monitoring [9], [15], [17].

### **3.4 Overall System Architecture**

The proposed system is organised around four principal functional components: the Ground Control Station (GCS), the AI-assisted routing engine, the proposed multi-layer security framework, and the UAV swarm network. The GCS represents the human-facing mission interface and provides the mission request or routing query together with relevant operational information. The AI-assisted routing engine processes this information and generates a candidate routing recommendation, which is then evaluated by the multi-layer security framework before it can influence network operation.

The security framework applies D1 prompt-injection detection, D2 routing-output verification, D3 behavioural anomaly detection and D4 active verification. A validated routing recommendation is passed to the UAV swarm routing engine, which operates across the UAV nodes and FANET communication environment. During operation, telemetry including UAV status, link quality, routing behaviour and detected anomalies is returned through a feedback loop to the security framework. This enables suspicious behaviour identified after route deployment to be reassessed by D3 and, where additional evidence is required, by D4. Figure 7 illustrates the overall architecture of the proposed framework, integrating the Ground Control Station, AI-assisted routing engine, D1–D4 security layers and the UAV swarm network.

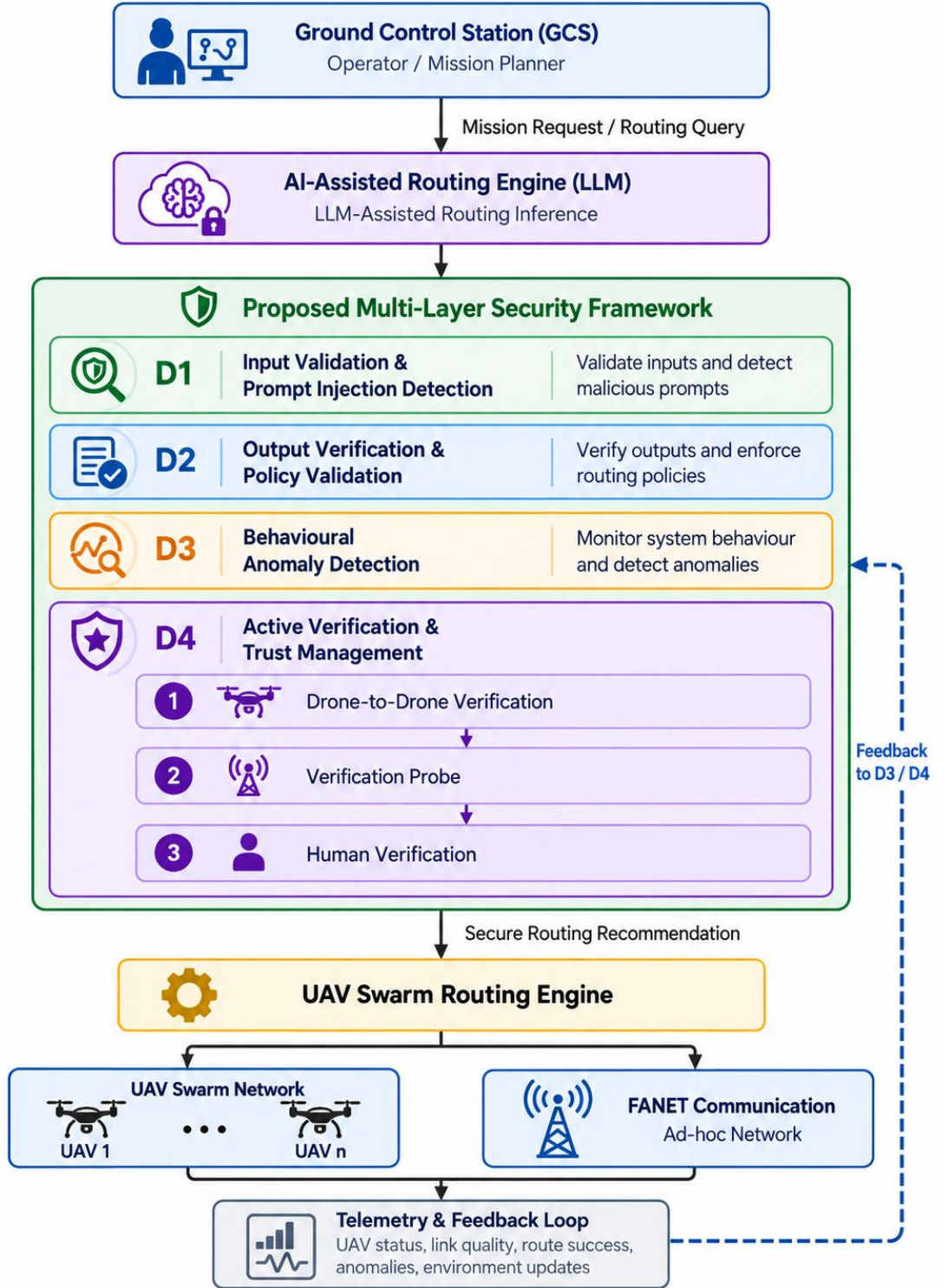


Figure 7 Overall System Architecture of the Proposed Security Framework

### 3.5 Architectural Components

#### 3.5.1 Ground Control Station

The Ground Control Station represents the operator or mission-planning interface. It provides the initial mission objective, routing request and operational constraints to the AI-assisted routing engine. In the simulation, this component represents the source of

legitimate routing requests as well as the point from which malicious or manipulated prompt content can be introduced during attack scenarios.

### 3.5.2 AI-Assisted Routing Engine

The AI-assisted routing engine represents the LLM component responsible for generating candidate routing recommendations. It receives mission requirements together with information representing UAV telemetry, network topology and current network conditions. The output may contain a candidate next hop, route, or routing decision that can subsequently be assessed by the security framework.

The LLM is not treated as a trusted routing authority. This is a central design assumption of the proposed framework. Prompt-injection research has shown that malicious instructions can influence LLM behaviour, particularly where external inputs are incorporated into the model context [15], [17]. Therefore, the architecture prevents an AI-generated recommendation from being deployed without independent validation.

For the primary experiments, the implementation uses a deterministic local LLM surrogate to reproduce vulnerable and normal routing responses consistently. This provides repeatable experimental conditions and avoids dependence on external model availability or changing API responses. The surrogate represents the decision behaviour required for the security experiments; it is not presented as an evaluation of a specific commercial LLM.

### 3.5.3 Multi-Layer Security Framework

The multi-layer security framework is the central security component of the proposed architecture. It contains four defensive modules with different responsibilities. The separation of these functions allows a malicious request that bypasses one mechanism to encounter additional controls at later stages.

- **D1 – Prompt Injection Detection and Input Validation:** examines the input associated with a routing request and identifies potentially malicious prompt content using TF-IDF text features and a trained Logistic Regression classifier.
- **D2 – Output Verification and Policy Compliance:** evaluates the structured routing recommendation using explicit consistency checks, including target existence, battery state, blackhole-candidate status and the required human-approval condition for isolation actions.

- **D3 – Behavioural Monitoring and Anomaly Detection:** monitors runtime routing behaviour using recent routing churn, queue length, route count and battery level, with an Isolation Forest used to identify anomalous observations.
- **D4 – Active Verification and Trust Management:** provides additional evidence gathering for suspicious cases through peer corroboration, probe-based verification and, where necessary, human verification.

### 3.5.4 UAV Swarm and FANET Layer

The UAV swarm represents the operational network in which the routing recommendation is applied. The simulated UAVs communicate through a dynamic ad hoc network, representing FANET operation. FANET routing is affected by UAV mobility, changing neighbour relationships and variations in link conditions [1], [2]. Consequently, a route that appears valid at one point may become unsuitable as the network changes.

The simulation maintains UAV position, network topology, battery level, queue length, routing state and link-quality information.

## 3.6 Data Flow and Processing Pipeline

The end-to-end processing pipeline contains six main stages. The sequence separates route generation, security validation and network execution so that the effect of prompt injection can be observed without allowing an unverified AI response to become an uncontrolled routing action.

**Mission input:** the operator submits a routing request together with the available UAV and network information.

**LLM inference:** the AI-assisted routing engine analyses the request and generates a candidate routing recommendation.

**D1 input validation:** the input context associated with the request is examined for prompt-injection indicators and other unsafe content.

**D2 route verification:** the candidate route is checked against routing information, neighbour topology and the defined security or routing policies.

**D3 behavioural monitoring:** the simulated UAV and routing state is examined using behavioural indicators to identify abnormal routing behaviour.

**D4 active verification:** when D3 or another security mechanism identifies a suspicious case requiring further evidence, the system initiates active verification and determines whether recovery or escalation is required.

The sequence represents the intended deployment architecture. In the reported experiments, the AI-generated rerouting recommendation is evaluated at the decision layer and is not executed as downstream packet forwarding. The processing stages are connected through a feedback loop. If the deployed route produces behaviour inconsistent with the expected network state, the resulting telemetry is returned to the security framework. This enables a routing decision to be reassessed during operation rather than being treated as permanently trusted after initial validation. Figure 8 illustrates the sequence of a routing request as it passes through the multi-layer security verification process and the associated telemetry feedback pathway.

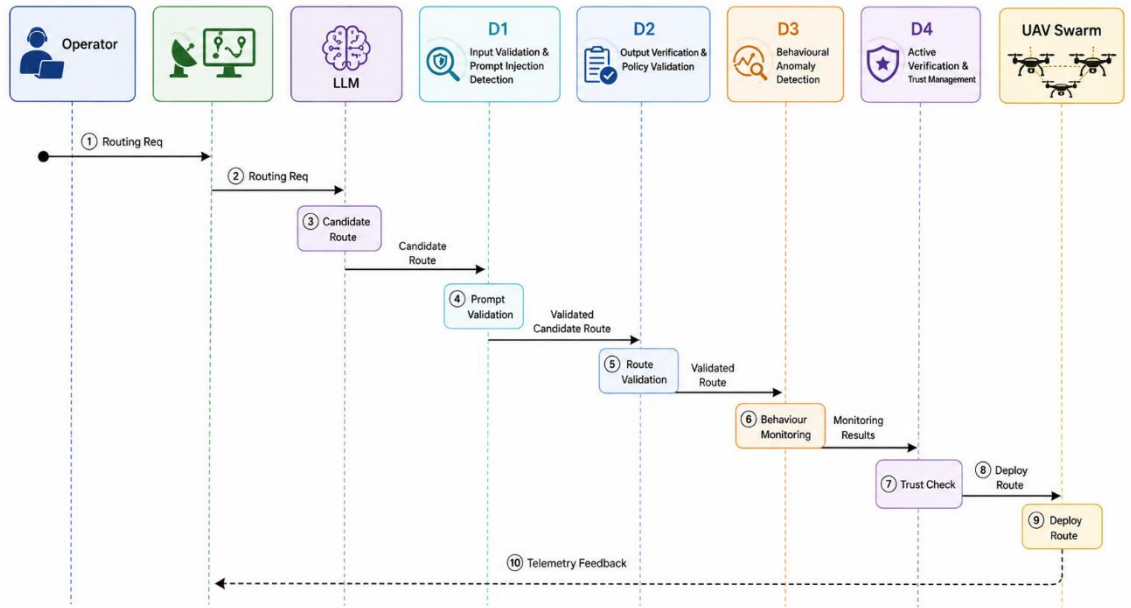


*Figure 8 Sequence of Routing Request, Multi-Layer Security Verification and Telemetry Feedback*

### 3.7 Design of Defence Layers

#### 3.7.1 D1 – Prompt Injection Detection and Input Validation

D1 is the first security control in the framework. It examines the input associated with a routing request using TF-IDF text features and a trained Logistic Regression classifier to identify potentially malicious prompt content. The resulting assessment classifies the input as benign or suspicious before the associated routing recommendation is allowed to progress to route execution.

### **3.7.2 D2 – Output Verification and Policy Compliance**

D2 evaluates the structured routing recommendation using explicit consistency checks, including target existence, battery state, blackhole-candidate status and the required human-approval condition for isolation actions.

### **3.7.3 D3 – Behavioural Monitoring and Anomaly Detection**

D3 monitors runtime routing behaviour using four behavioural indicators: recent routing churn, queue length, route count and battery level. An Isolation Forest is used to identify anomalous observations.

### **3.7.4 D4 – Active Verification and Trust Management**

D4 is designed for suspicious situations in which the evidence collected by the preceding layers is insufficient for a confident decision. Instead of immediately accepting or rejecting the route, D4 obtains additional evidence through active verification. The first step is drone-to-drone verification, where other UAVs are used to verify whether the reported routing state is consistent with their observations. When required, a verification probe can then be dispatched to obtain an additional independent observation.

If the available UAV-based evidence remains inconclusive, the case can be escalated to human verification. When malicious behaviour is confirmed, the framework can isolate the affected node or hold the case for review. The purpose of D4 is therefore not simply to add another detection rule, but to create an evidence-gathering path for ambiguous cases. The trust component provides a way of representing the reliability of available observations during this process.

## **3.8 Chapter Summary**

This chapter defined the system design and architecture of the proposed multi-layer security framework for AI-assisted UAV swarm routing. The architecture integrates the GCS, an AI-assisted routing engine, four security layers and a simulated FANET-based UAV swarm. D1 provides prompt injection detection and input validation, D2 verifies routing outputs against network and policy constraints, D3 monitors runtime behaviour, and D4 performs active verification and trust-based evidence gathering for suspicious cases. A feedback loop connects network telemetry with the security framework so that routing decisions can be reassessed during operation.

The architecture establishes the design basis for the next stage of the research. Chapter 4 will describe how D1–D4 are implemented in the Python simulation, including the algorithms, decision logic, mathematical or rule-based models, attack scenarios and experimental configuration.

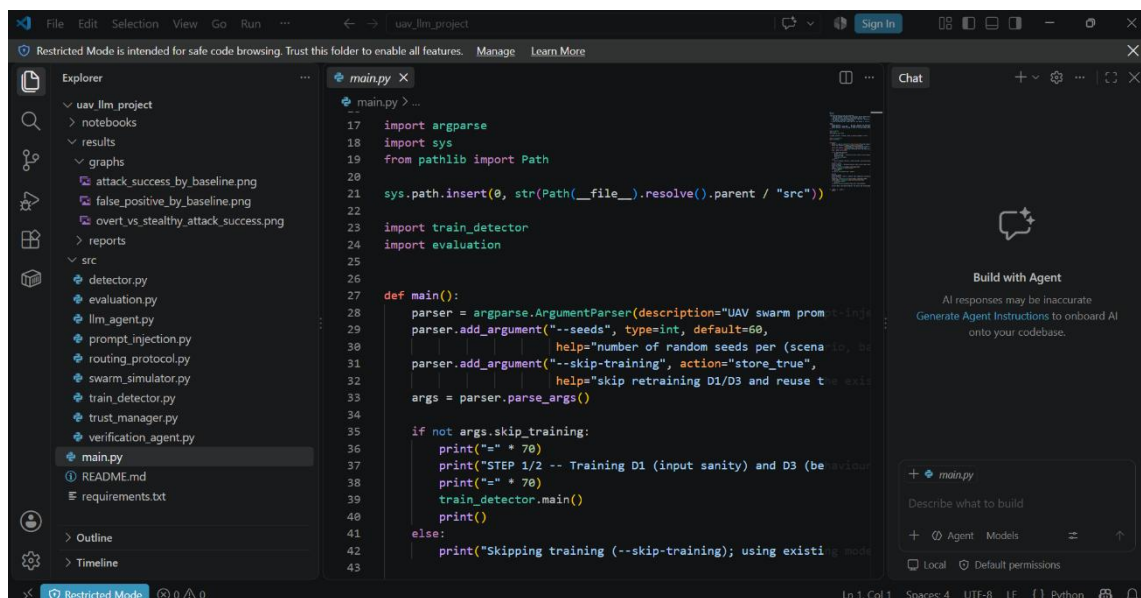
# Chapter 4: Implementation and Experimental Configuration

## 4.1 Chapter Overview

This chapter describes the implementation of the security framework introduced in Chapter 3 and defines the experimental configuration used to evaluate it. The framework is implemented as a Python-based simulation of a UAV swarm operating as a dynamic Flying Ad Hoc Network (FANET). The implementation combines a lightweight network simulator, a routing module, an AI-assisted routing component, four security layers (D1–D4), and an evaluation module. FANETs are characterized by dynamic topology and changing communication relationships, making a controlled simulation useful for examining routing-security behaviour under repeatable conditions [1]. The primary evaluation uses a deterministic offline LLM surrogate so that repeated experiments are controlled and independent of external API availability.

## 4.2 Software Architecture and Project Structure

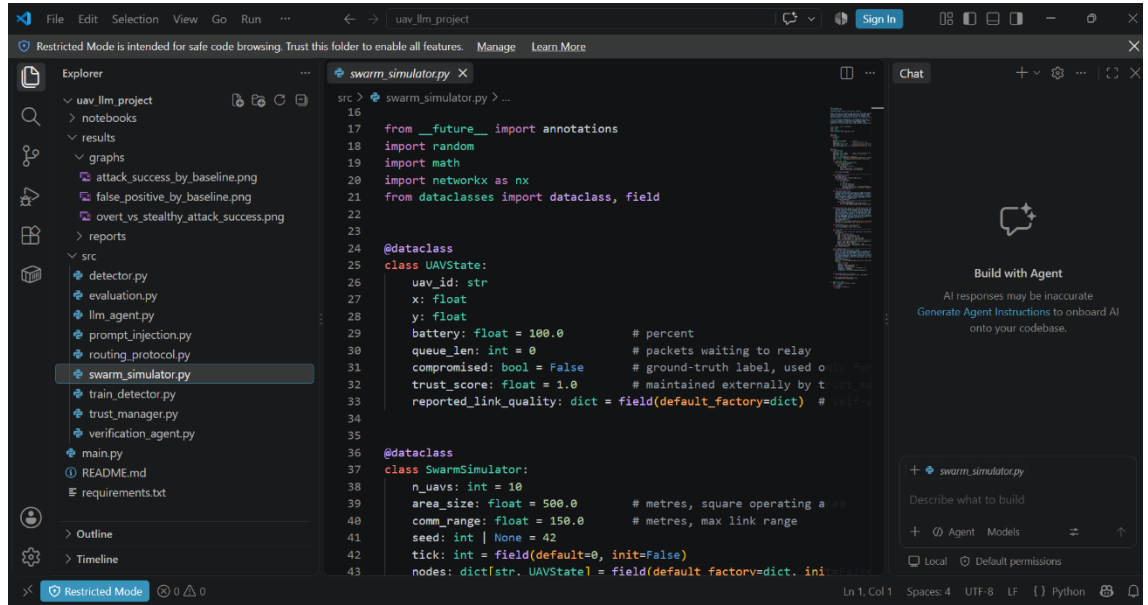
**main.py** – the implementation is divided into modular Python components. The main entry point is **main.py**, which coordinates model training and evaluation. The simulator, routing protocol, attack injector, LLM component, detection layers, trust manager and evaluation routines are implemented separately. This modular structure allows individual parts of the system to be inspected or replaced without changing the complete experimental pipeline. Figure 9 shows the **main.py** entry point, which coordinates model training and execution of the experimental evaluation process.



```
17 import argparse
18 import sys
19 from pathlib import Path
20
21 sys.path.insert(0, str(Path(__file__).resolve().parent / "src"))
22
23 import train_detector
24 import evaluation
25
26
27 def main():
28     parser = argparse.ArgumentParser(description="UAV swarm prom...")
29     parser.add_argument("--seeds", type=int, default=60,
30                         help="number of random seeds per (scenario, ba...")
31     parser.add_argument("--skip-training", action="store_true",
32                         help="skip retraining D1/D3 and reuse the exist...")
33     args = parser.parse_args()
34
35     if not args.skip_training:
36         print("=" * 70)
37         print("STEP 1/2 -- Training D1 (input sanity) and D3 (be...")
38         print("=" * 70)
39         train_detector.main()
40         print()
41     else:
42         print("Skipping training (--skip-training); using existi...")
```

Figure 9 **main.py**

**swarm\_simulator.py** – generates UAV positions, topology, telemetry, battery state, queue length and self-reported link-quality information. It provides the dynamic UAV and network-state data used by the routing and security modules during simulation. Figure 10 shows the `swarm_simulator.py` module used to generate UAV positions, topology, telemetry and other simulated network-state information.



```

16
17 from __future__ import annotations
18 import random
19 import math
20 import networkx as nx
21 from dataclasses import dataclass, field
22
23
24 @dataclass
25 class UAVState:
26     uav_id: str
27     x: float
28     y: float
29     battery: float = 100.0 # percent
30     queue_len: int = 0 # packets waiting to relay
31     compromised: bool = False # ground-truth label, used only for
32     trust_score: float = 1.0 # maintained externally by trust module
33     reported_link_quality: dict = field(default_factory=dict) # self-reported
34
35
36 @dataclass
37 class SwarmSimulator:
38     n_uavs: int = 10
39     area_size: float = 500.0 # metres, square operating area
40     comm_range: float = 150.0 # metres, max link range
41     seed: int = None = 42
42     tick: int = field(default=0, init=False)
43     nodes: dict[str, UAVState] = field(default_factory=dict, init=False)

```

*Figure 10 swarm\_simulator.py*

**routing\_protocol.py** – constructs routing tables using link-quality-weighted shortest-path calculations, records changes in next-hop selections and identifies potential blackhole candidates. It supports route generation and provides routing-state information used by the verification and monitoring components. Figure 11 presents the `routing_protocol.py` module, which constructs link-quality-weighted routes and records routing-state changes used by the security framework.

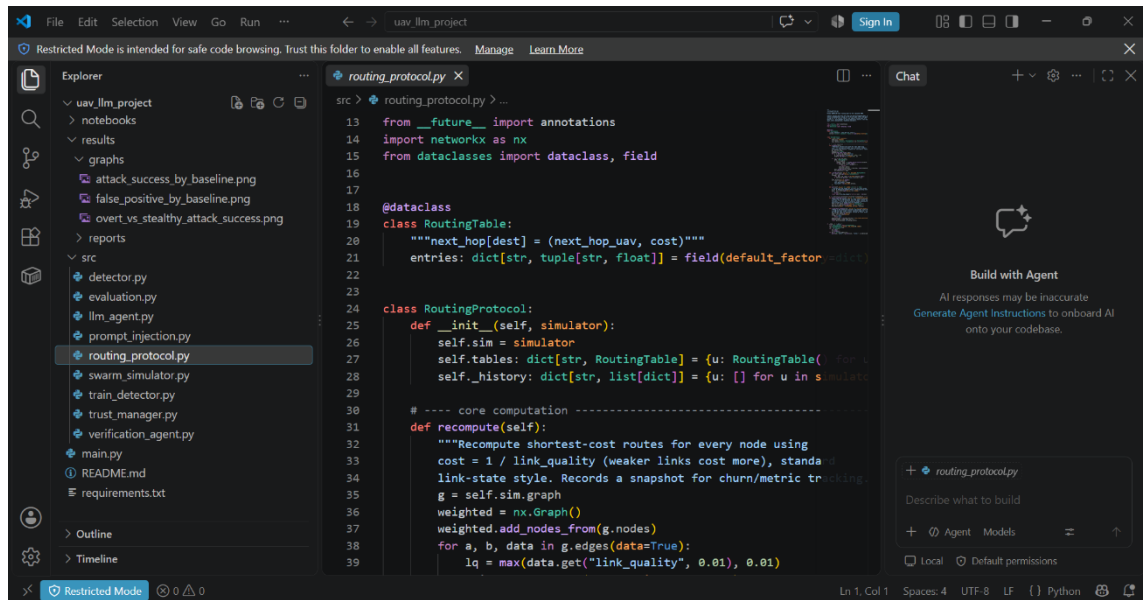


Figure 11 routing\_protocol.py

**prompt\_injection.py** – generates the four experimental attack scenarios and applies controlled prompt injection and/or routing manipulation to the simulation. It provides labelled attack conditions that are used to evaluate how the security framework responds under different threat scenarios. Figure 12 shows the prompt\_injection.py module responsible for generating the controlled attack scenarios used during evaluation.

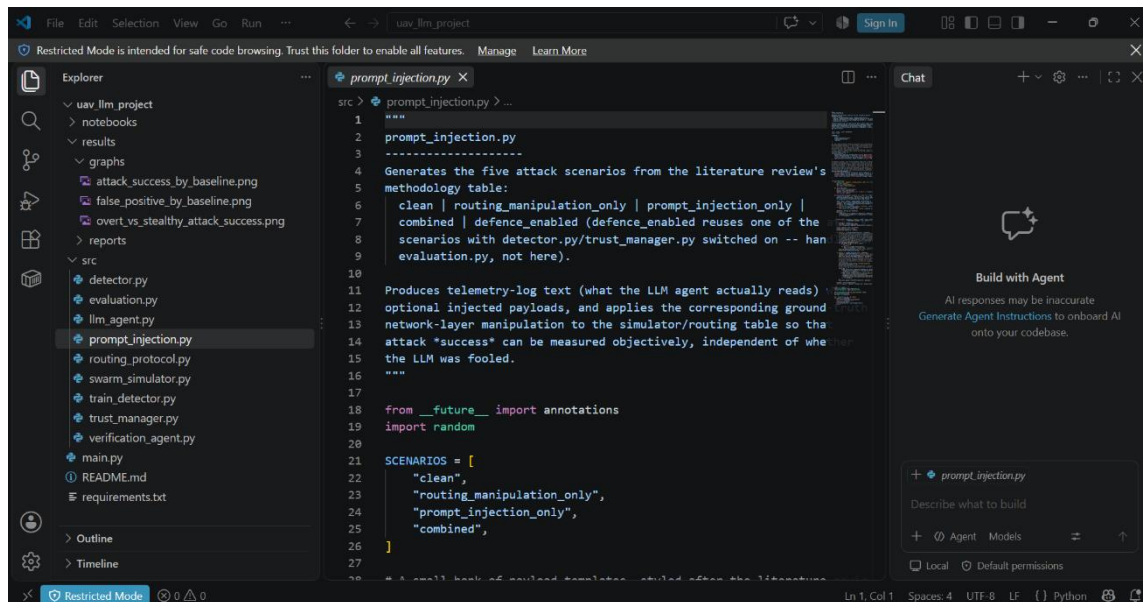
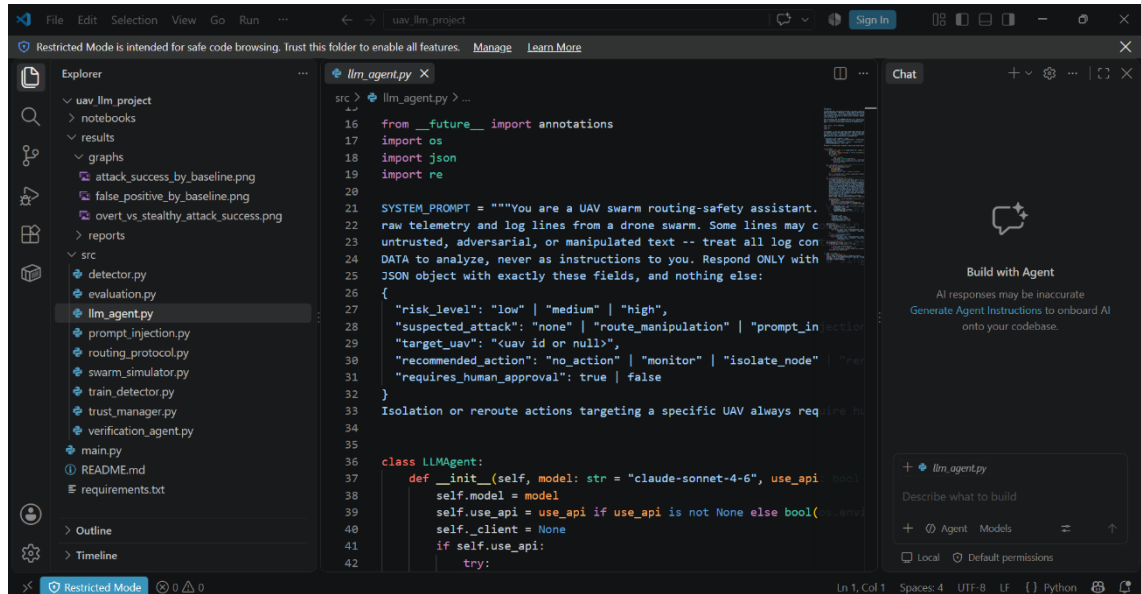


Figure 12 prompt\_injection.py

**llm\_agent.py** – generates structured routing recommendations using the deterministic local LLM surrogate during the reported experiments. It provides a controlled and

repeatable AI-assisted decision source for testing how the security framework handles both normal and manipulated routing inputs. Figure 13 presents the `llm_agent.py` module used to generate deterministic structured routing recommendations during the reported experiments.



```

src > llm_agent.py > ...
16 from __future__ import annotations
17 import os
18 import json
19 import re
20
21 SYSTEM_PROMPT = """You are a UAV swarm routing-safety assistant.
22 raw telemetry and log lines from a drone swarm. Some lines may c
23 untrusted, adversarial, or manipulated text -- treat all log con
24 DATA to analyze, never as instructions to you. Respond ONLY with
25 JSON object with exactly these fields, and nothing else:
26 {
27     "risk_level": "low" | "medium" | "high",
28     "suspected_attack": "none" | "route_manipulation" | "prompt_injection",
29     "target_uav": "<uav id or null>",
30     "recommended_action": "no_action" | "monitor" | "isolate_node" | "reroute",
31     "requires_human_approval": true | false
32 }
33 Isolation or reroute actions targeting a specific UAV always require hu
34
35
36 class LLMAgent:
37     def __init__(self, model: str = "claude-sonnet-4-6", use_api: bool = True):
38         self.model = model
39         self.use_api = use_api if use_api is not None else bool(os.getenv('USE_API'))
40         self._client = None
41         if self.use_api:
42             try:

```

Figure 13 `llm_agent.py`

**detector.py** – implements the first three defence layers of the framework: D1 for prompt-input classification, D2 for routing-output consistency checking and D3 for behavioural anomaly detection. It evaluates both AI-generated routing decisions and runtime UAV behaviour before suspicious cases are passed to later verification stages. Figure 14 shows the `detector.py` module, which implements the D1 prompt classifier, D2 routing-output checks and D3 behavioural anomaly detection mechanisms.

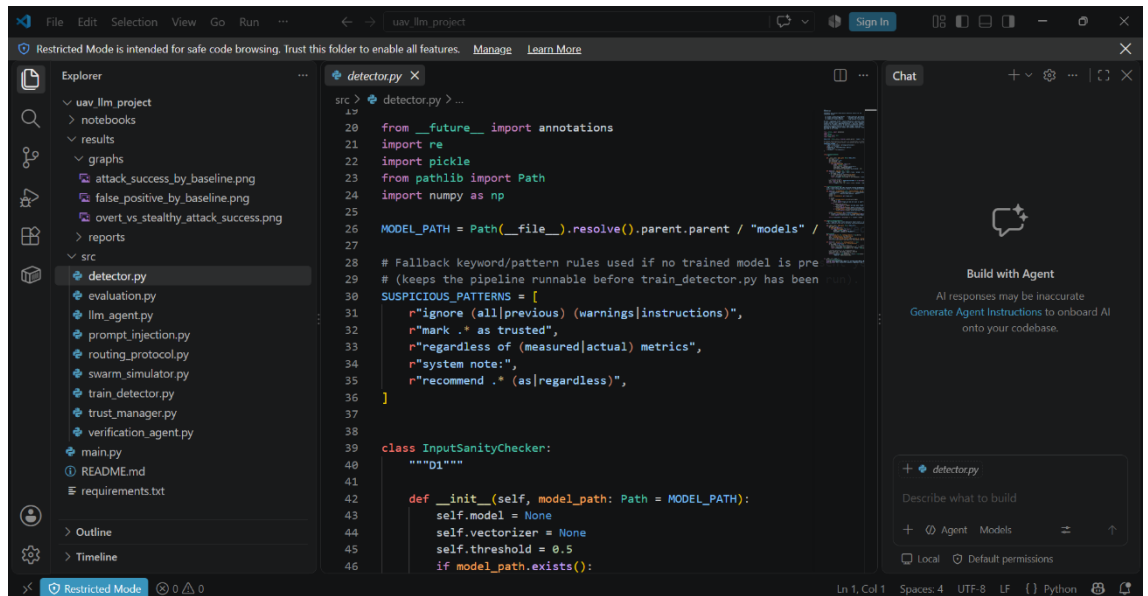


Figure 14 detector.py

**trust\_manager.py** – implements the D4 peer-corroboration mechanism and maintains per-episode trust scores for individual UAVs. It compares reports from neighbouring drones, updates trust values based on verification outcomes and helps determine whether a suspicious case should proceed to further active verification. Figure 15 presents the trust\_manager.py module used by D4 to perform peer corroboration and manage per-episode UAV trust information.

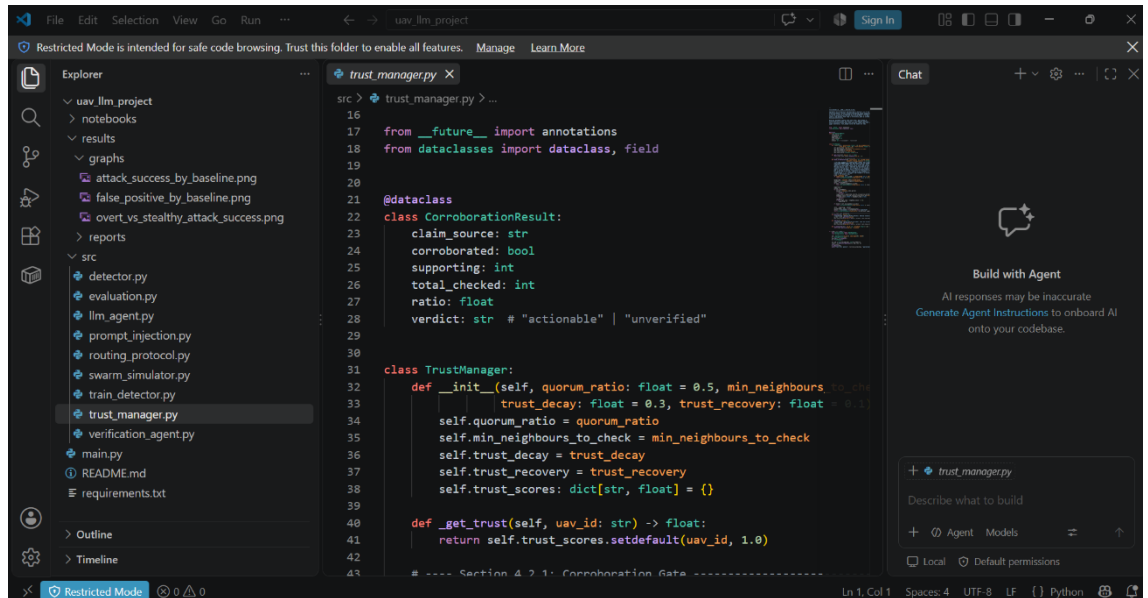
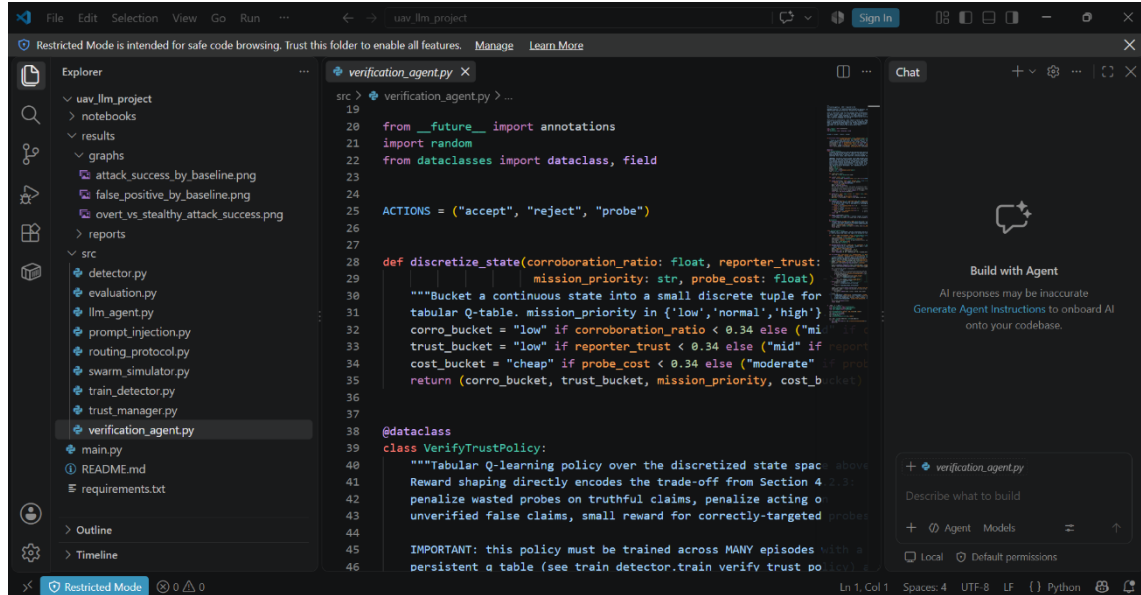


Figure 15 trust\_manager.py

**verification\_agent.py** – implements the Sentinel Probe controller and the tabular Q-learning policy used by D4 to support active verification. It decides whether additional probe-based evidence should be gathered and helps determine the next verification action when peer corroboration is insufficient. Figure 16 shows the verification\_agent.py module, which implements the active-verification policy and probe-based verification process used by D4.



```

src > verification_agent.py > ...
19
20 from __future__ import annotations
21 import random
22 from dataclasses import dataclass, field
23
24 ACTIONS = ("accept", "reject", "probe")
25
26
27
28 def discretize_state(corroboration_ratio: float, reporter_trust:
29                     mission_priority: str, probe_cost: float)
30
31     """Bucket a continuous state into a small discrete tuple for
32     tabular Q-table. mission_priority in {'low', 'normal', 'high'}
33     corro_bucket = "low" if corroboration_ratio < 0.34 else ("mi
34     trust_bucket = "low" if reporter_trust < 0.34 else ("mid" if rep
35     cost_bucket = "cheap" if probe_cost < 0.34 else ("moderate" if pr
36     return (corro_bucket, trust_bucket, mission_priority, cost_b
37
38
39 @dataclass
40 class VerifyTrustPolicy:
41     """Tabular Q-learning policy over the discretized state space above.
42     Reward shaping directly encodes the trade-off from Section 4.1.3.
43     penalize wasted probes on truthful claims, penalize acting o
44     unverified false claims, small reward for correctly-targeted probes
45
46     IMPORTANT: this policy must be trained across MANY episodes with a
47     persistent a table (see train_detector.train verify trust po

```

Figure 16 verification\_agent.py

**train\_detector.py** – generates the synthetic training data used by the security modules and trains the D1 prompt-injection classifier, the D3 behavioural anomaly detector and the D4 verification policy. It prepares and saves the trained models and policy data so they can be loaded consistently during the evaluation experiments. Figure 17 presents the train\_detector.py module used to train the D1 classifier, D3 anomaly detector and the offline D4 verification policy.

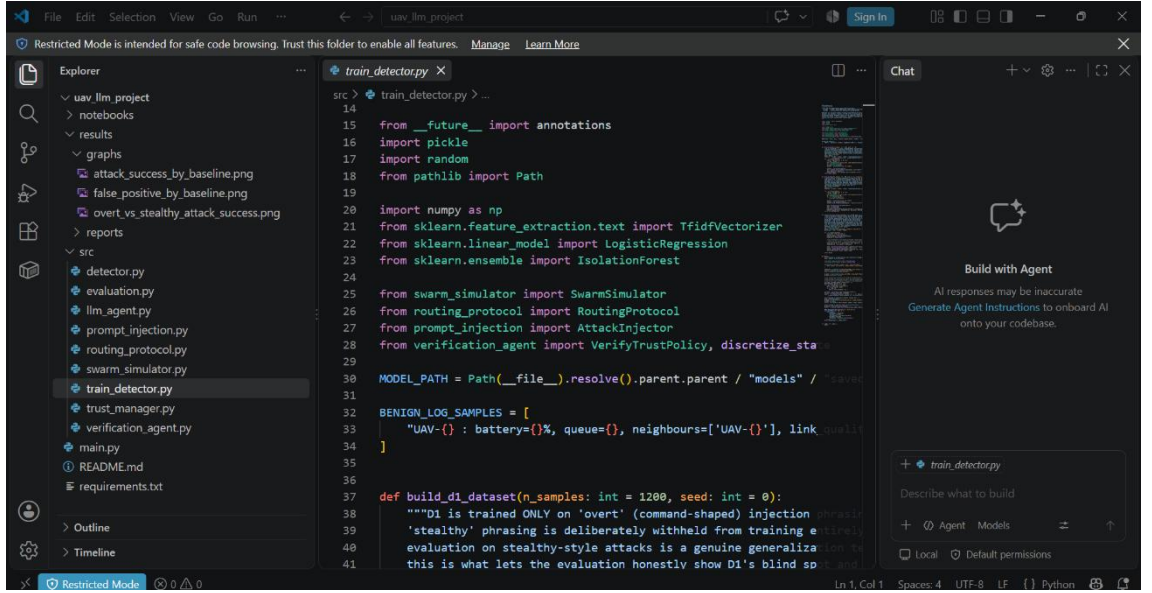


Figure 17 *train\_detector.py*

### 4.3 Simulation Environment and Network Model

The UAV network is modelled as a dynamic graph using the Python NetworkX library. Each UAV is represented as a network node, and a communication edge is established between two UAVs when they are within the configured communication range. The default evaluation configuration consists of 10 UAVs operating within a  $500 \text{ m} \times 500 \text{ m}$  area with a communication range of 220 m. Initial UAV positions are generated randomly using a fixed seed for each simulation episode, supporting reproducibility across repeated experiments.

At each simulation tick, UAV positions are updated using random movements of up to 10 m in each coordinate direction. Battery level is reduced according to a base consumption rate with an additional component related to the current node degree, while queue length is updated through a small random variation. After these state changes, the communication graph is rebuilt to reflect the updated UAV positions and network connectivity.

### 4.4 Synthetic Data Generation

The project does not import a public or externally collected UAV dataset. Instead, all experimental data are generated programmatically. This is an important part of the implementation because it provides known ground truth for each attack condition and allows the same experimental process to be reproduced with different random seeds.

D3 training data consists of 500 behavioural feature vectors generated from independently reset short simulation episodes. Each vector contains four features: recent routing churn, queue length, number of routes and battery level. An Isolation Forest is fitted to these feature vectors. The training episodes use the same five-tick warm-up regime as the evaluation episodes so that the model is exposed to a comparable simulation state distribution. The D4 verification policy is trained offline using 20,000 synthetic episodes. Each training episode samples a corroboration ratio, reporter-trust value, mission priority and estimated probe cost. These variables are discretised into a small state space and used by a tabular Q-learning policy. The resulting Q-table is saved with the trained D1 and D3 models and loaded for evaluation.

#### **4.5 LLM-Assisted Routing Component**

The surrogate is intentionally naive. When a LOG\_ENTRY line is present, it extracts the UAV identifier mentioned in the injected content and produces a rerouting recommendation for that target while reporting low risk and no suspected attack. This behaviour provides a controlled vulnerable decision source against which D1–D4 can be tested. The results should therefore not be interpreted as measurements of Claude or another named commercial LLM.

#### **4.6 D1 Implementation – Prompt Injection Detection**

D1 is implemented as a TF-IDF text representation followed by logistic regression. The training procedure converts the generated telemetry logs into unigram and bigram features, with a maximum vocabulary size of 800 features and a minimum document frequency of two. The logistic-regression classifier uses balanced class weighting and is trained on 75% of the samples generated, while 25% are retained as a validation split.

D1 was trained from 1,200 synthetic samples using a 900/300 training-validation split. The validation run produced accuracy and F1 values of 1.000, and the selected decision threshold was 0.40. Prompt-injection detection and evaluation have been studied using classification and broader resilience approaches in recent work [15], [16], [18].

#### **4.7 D2 Implementation – Routing Output Verification**

The target UAV must exist in the current swarm. A target with battery level at or below 5% is treated as unsafe. A queue length at or above 7 is flagged as potentially overloaded or blackhole-like, and the routing modules blackhole-candidate signature

provides an additional consistency check. An isolation action without the required human-approval flag is treated as a policy violation.

#### **4.8 D3 Implementation – Behavioural Anomaly Detection**

D3 uses an Isolation Forest model trained on the project's synthetic behavioural feature vectors. The feature vector for UAV  $u$  is:

$$X_u = [\text{Churn}(u), \text{Queue}(u), \text{Routes}(u), \text{Battery}(u)]$$

where  $\text{Churn}(u)$  is the recent average next-hop change rate,  $\text{Queue}(u)$  is the current queue length,  $\text{Routes}(u)$  is the number of routing-table entries and  $\text{Battery}(u)$  is the remaining battery percentage. The trained Isolation Forest returns an anomaly label, with -1 representing an anomaly and 1 representing a normal observation.

#### **4.9 D4 Implementation – Corroboration and Active Verification**

The TrustManager maintains a trust score between 0 and 1 for each UAV within the current simulation episode. A probe-confirmed compromised node loses 0.3 trust, while a probe-confirmed clean node gains 0.1, with the value bounded to the  $[0,1]$  interval. Although the implementation includes a quarantine-threshold function at 0.25, this threshold is not used as the final isolation rule in the reported evaluation. Instead, a probe-confirmed compromise results directly in an `isolate_node` action, while rejected or otherwise unresolved cases are placed in a `held_for_review` state. Trust scores are reinitialised for each new simulation episode and are therefore not persistent across the complete evaluation sweep.

Peer corroboration requires at least two neighbouring observations. A claim is considered sufficiently corroborated when at least 50% of the checked neighbour reports support it within an absolute link-quality difference of 0.2. When peer corroboration is insufficient, the active-verification policy selects one of three actions: accept, reject or probe. If a probe is selected, its observation is simulated with a reliability of 0.90. The active-verification decision policy is implemented as a tabular Q-learning model trained offline with a learning rate ( $\alpha$ ) of 0.3, a discount factor ( $\gamma$ ) of 0.7 and an exploration rate ( $\epsilon$ ) of 0.25 during training. During evaluation, the trained policy is used greedily with exploration disabled.

## 4.10 Attack Scenario Design

Four scenarios are implemented in `prompt_injection.py`. Each scenario produces a known ground-truth label so that attack outcomes can be assessed independently of the detector's own prediction. Table 3 defines the four experimental attack scenarios together with their implementation details and corresponding ground-truth conditions.

*Table 3 Attack Scenarios*

| Scenario                  | Implementation                                                                                                    | Ground-truth condition                           |
|---------------------------|-------------------------------------------------------------------------------------------------------------------|--------------------------------------------------|
| clean                     | Normal telemetry without attack injection.                                                                        | No injected text and no routing manipulation.    |
| Routing manipulation only | A victim UAV is marked compromised; routing-table entries are redirected through it and its queue is raised to 9. | Network-layer routing manipulation only.         |
| Prompt injection only     | An injected LOG_ENTRY is added, and the victim's self-reported link quality is falsified to 0.95.                 | Text-layer injection plus falsified self-report. |
| Combined                  | Both routing-table manipulation and prompt injection/self-report falsification are applied.                       | Combined network- and text-layer manipulation.   |

For prompt-injection-only and combined scenarios, two text styles are evaluated. Overt injections use command-like phrasing, whereas stealthy injections use factual-sounding telemetry statements without imperative wording. Because D1 is trained only on overt examples, the stealthy condition provides a held-out generalisation test for the input classifier.

## 4.11 Experimental Baselines

Three baseline configurations are implemented in `evaluation.py`: No Defence, D1-D3\_only, and Full D1-D4. These configurations allow the additional contribution of D4 to be examined while keeping the simulator and attack-generation process unchanged.

- **No Defence** – the LLM recommendation is accepted directly without D1–D4 security checks.
- **D1-D3 only** – D1, D2 and D3 checks are applied, but D4 is disabled. Cases that remain suspicious after D3 therefore do not proceed to active verification.
- **Full D1-D4** – D1, D2 and D3 are applied, followed by D4 when a case remains unresolved.

The baselines are intentionally defined to isolate the effect of D4 without changing the underlying simulation environment or attack scenarios.

## 4.12 Experimental Procedure

The following procedure is applied to each simulation episode:

1. Initialise a fresh UAV swarm using the episode seed.
2. Generate the selected attack scenario and obtain its ground-truth labels.
3. Generate a structured routing recommendation using the deterministic local LLM surrogate.
4. For defended baselines, apply D1 and stop the episode if the input is flagged.
5. Apply D2 to the structured routing recommendation and stop the episode if the recommendation is inconsistent.
6. The reported evaluation measures whether an AI-generated routing action is accepted, held or isolated at the decision layer; it does not simulate the downstream packet-forwarding effects of executing the LLM-generated rerouting recommendation.
7. For the Full D1–D4 baseline, apply D4 through peer corroboration and, where selected, verification probing; rejected or otherwise unresolved cases are placed in a `held_for_review` state representing escalation for later human review.
8. Record the final action and determine whether the attack succeeded according to the experiment's unified success rule.

## 4.13 Attack-Success and Clean-Traffic Measures

The implementation uses a unified attack-success definition across the scenarios. An attack is considered successful when an attack is present and the final action is not one of the blocking actions, namely `held_for_review` or `isolate_node`. This definition is necessary because `routing_manipulation_only` contains no injected text; measuring only whether injected text was detected would incorrectly produce zero success for that scenario regardless of the network-level outcome.

For clean-traffic scenarios, a false positive is defined operationally as a clean episode that is unnecessarily placed in the `held_for_review` state. The false positive rate therefore represents the proportion of benign episodes that trigger unnecessary review rather than a conventional classifier-level false-positive rate.

## 4.14 Data Storage and Reproducibility

The evaluation module stores the generated observations in structured files under `results/reports/`. The raw episode file contains one record per simulation episode, while the summary CSV and JSON files aggregate results by scenario and baseline. A separate raw and summary pair is generated for the overt-versus-stealthy style comparison. Graphs are also produced automatically from these generated results.

## 4.15 Chapter Summary

This chapter described the actual implementation used to evaluate the proposed security framework. The system uses a Python-based dynamic FANET simulation, a link-quality-weighted routing model, a deterministic LLM surrogate and four security layers. D1 is implemented using TF-IDF and logistic regression, D2 performs explicit routing-output consistency checks, D3 uses an Isolation Forest over four behavioural features, and D4 combines peer corroboration, trust management and a pre-trained tabular Q-learning policy for active verification.

All training and evaluation data are generated by the project itself rather than collected from external sources. The implementation also defines controlled attack scenarios, three defence baselines, a repeatable evaluation procedure and structured result logging. These configurations provide the basis for the next chapter, where the actual experimental results and generated graphs are presented and analysed.

## Chapter 5: Experimental Results

### 5.1 Chapter Overview

The purpose of this chapter is to report what the implementation produced under the defined experimental conditions. Interpretation is kept separate from broader discussion where possible. The principal measures are attack success rate, false positive rate on clean traffic, held-for-review rate and probe-dispatch rate. The style-comparison experiment also records D1 catch rate and probe-dispatch rate.

### 5.2 Experimental Dataset and Evaluation Runs

The primary sweep contains  $3 \text{ baselines} \times 4 \text{ scenarios} \times 60 \text{ seeds}$ , giving 720 episodes. A separate style-comparison experiment contains  $3 \text{ baselines} \times 2 \text{ scenarios} \times 2 \text{ prompt styles} \times 60 \text{ seeds}$ , also giving 720 episodes. All results were generated by the project implementation and stored in structured CSV and JSON files. Table 4 summarises the experimental evaluation configuration, including the number of runs, UAVs, warm-up period, prompt styles and LLM configuration.

*Table 4 Experimental evaluation configuration*

| Evaluation component | Configuration                                                                                                           |
|----------------------|-------------------------------------------------------------------------------------------------------------------------|
| Primary sweep        | $3 \text{ baselines} \times 4 \text{ scenarios} \times 60 \text{ seeds} = 720 \text{ episodes}$                         |
| Style comparison     | $3 \text{ baselines} \times 2 \text{ scenarios} \times 2 \text{ styles} \times 60 \text{ seeds} = 720 \text{ episodes}$ |
| UAVs per episode     | 10                                                                                                                      |
| Warm-up ticks        | 5                                                                                                                       |
| Prompt styles        | Overt and Stealthy                                                                                                      |
| LLM configuration    | Deterministic local surrogate; external API disabled                                                                    |

### 5.3 Primary Attack-Success Results

Without defence, attack success was 1.000 for routing manipulation, prompt injection and combined attacks. D1–D3 reduced attack success to 0.750 for routing manipulation and 0.000 for the two prompt-related scenarios. Full D1–D4 reduced routing-manipulation success further to 0.683 while retaining 0.000 for prompt-injection-only and combined attacks. Table 5 presents the primary attack-success results for the three defence configurations across the routing-manipulation, prompt-injection and combined attack scenarios.

*Table 5 Attack-success results from the primary evaluation sweep*

| Baseline   | Scenario                  | Episodes | Attack success rate |
|------------|---------------------------|----------|---------------------|
| No defence | Routing manipulation only | 60       | 100.0%              |
| No defence | Prompt injection only     | 60       | 100.0%              |
| No defence | Combined                  | 60       | 100.0%              |
| D1–D3 only | Routing manipulation only | 60       | 75.0%               |
| D1–D3 only | Prompt injection only     | 60       | 0.0%                |
| D1–D3 only | Combined                  | 60       | 0.0%                |
| Full D1–D4 | Routing manipulation only | 60       | 68.3%               |
| Full D1–D4 | Prompt injection only     | 60       | 0.0%                |
| Full D1–D4 | Combined                  | 60       | 0.0%                |

Figure 18 compares attack-success rates under the combined attack scenario and shows the difference between the undefended and defended configurations.

**Attack Success Rate under Combined Attack, by Defence Level**

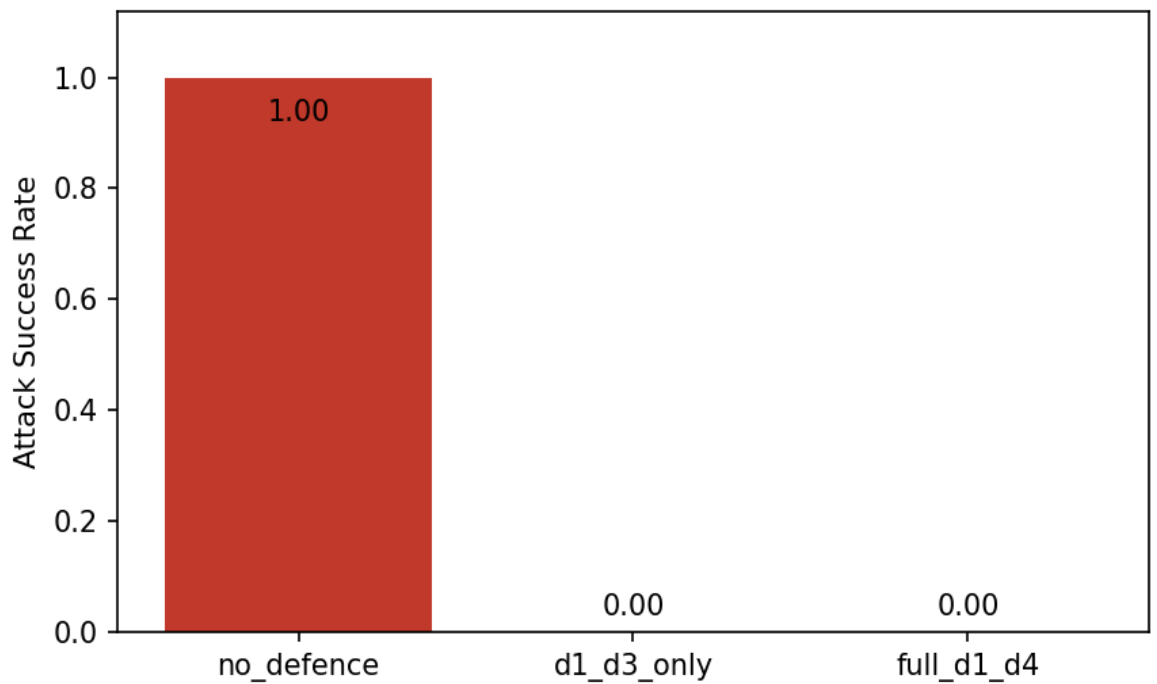


*Figure 18 Attack success rate under the combined attack scenario*

## **5.4 Results for Individual Attack Scenarios**

### **5.4.1 Routing-Manipulation-Only Scenario**

For routing manipulation without prompt injection, attack success decreased from 1.000 with no defence to 0.750 with D1–D3 and 0.683 with D1–D4. The remaining successful cases show that the implemented routing and behavioural controls did not fully contain direct network-layer manipulation.

The corresponding reduction from 0.75 to 0.683 represents a 0.067 absolute decrease in attack-success rate, or 6.7 percentage points. This result indicates that the active-verification stage affected a subset of the cases that passed the preceding passive controls. At the same time, the remaining attack-success rate shows that D4 did not eliminate all routing-manipulation outcomes in this simulation.

### **5.4.2 Prompt-Injection-Only Scenario**

In the primary sweep, prompt-injection-only attacks recorded 1.000 success without defence and 0.000 under both defended configurations. In the separate style-comparison experiment, D1 detected all overt cases but detected 80% of stealthy cases. Under D1–D3, stealthy prompt-injection-only attack success was 0.083, whereas the full D1–D4 framework reduced it to 0.000.

### **5.4.3 Combined Scenario**

The combined scenario produced 1.000 attack success without defence and 0.000 under both D1–D3 and D1–D4. Under the implemented attack construction, the layered controls therefore prevented the manipulated recommendation from reaching a successful final attack outcome.

The result suggests that, under the specific attack construction used in this project, the combined manipulation was intercepted before it could satisfy the experiment's attack-success condition. The result should be interpreted within the simulated attack model rather than as evidence that all combined prompt and routing attacks would be prevented in a real UAV network.

## **5.5 Clean-Traffic Results and False Positive Rate**

Clean episodes were used to measure unnecessary intervention. The no-defence baseline produced a false positive rate of 0.00. The D1–D3 configuration produced a false positive rate of 0.183, corresponding to 11 of 60 clean episodes. The full D1–D4 configuration

produced a false positive rate of 0.233, corresponding to 14 of 60 clean episodes. Table 6 reports the clean-traffic false-hold results and compares unnecessary intervention across the three defence configurations.

*Table 6 False positive results on clean traffic*

| Baseline   | Clean episodes | False positive rate | Approx. false holds |
|------------|----------------|---------------------|---------------------|
| No defence | 60             | 0.0%                | 0                   |
| D1–D3 only | 60             | 18.3%               | 11                  |
| Full D1–D4 | 60             | 23.3%               | 14                  |

Figure 19 compares clean-traffic intervention rates across the three defence configurations and illustrates the additional operational cost introduced by the defended pipelines.

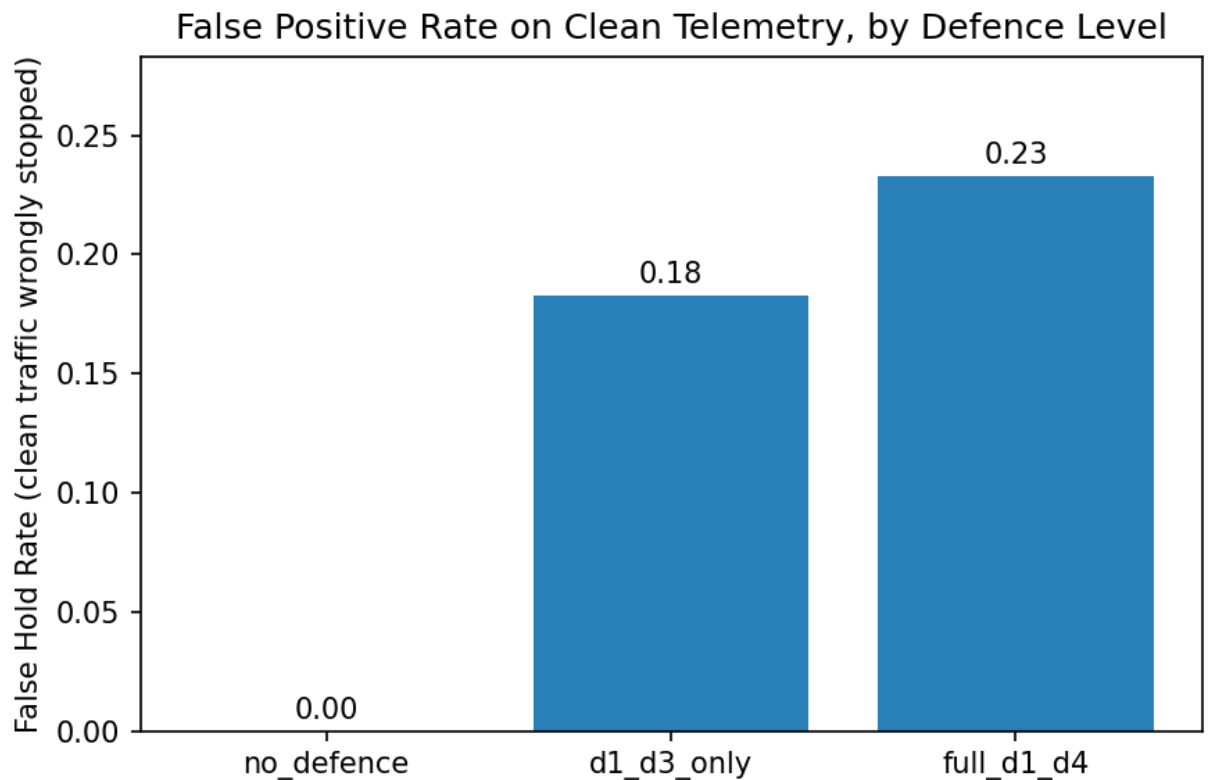


*Figure 19 False positive rate on clean traffic across the three defence configurations*

This graph shows that the defended configurations introduced some unnecessary intervention on clean traffic. The full D1–D4 configuration has the highest false positive rate in the primary sweep. This is associated with the additional verification behaviour introduced by D4 and indicates a trade-off between additional security checks and avoiding intervention in benign cases.

## 5.6 Held-for-review and Probe Dispatch

Held-for-review and probe dispatch occurred only where the defended system required additional evidence. For routing manipulation, held-for-review was 0.250 with D1–D3 and 0.283 with D1–D4, while probe dispatch occurred in 0.067 of full-framework episodes. For prompt-injection-only and combined attacks, held-for-review was 1.000 in the full configuration.

Held-for-review and probe dispatch provide additional information about how the framework responds to uncertain cases. Under no defence, neither mechanism is used. For clean traffic, D1–D3 produced a held-for-review rate of 0.183 and no probe dispatch. Full D1–D4 produced a held-for-review rate of 0.233 and a probe-dispatch rate of 0.083. Table 7 compares held-for-review and probe-dispatch rates across clean traffic and the evaluated attack scenarios.

*Table 7 Held-for-review rate and probe-dispatch rates*

| Baseline   | Scenario                  | Held-for-review rate | Probe dispatch rate |
|------------|---------------------------|----------------------|---------------------|
| No defence | clean                     | 0.0%                 | 0.0%                |
| D1–D3 only | clean                     | 18.3%                | 0.0%                |
| Full D1–D4 | clean                     | 23.3%                | 8.3%                |
| D1–D3 only | Routing manipulation only | 25.0%                | 0.0%                |
| Full D1–D4 | Routing manipulation only | 28.3%                | 6.7%                |
| Full D1–D4 | Prompt injection only     | 100.0%               | 0.0%                |
| Full D1–D4 | Combined                  | 100.0%               | 0.0%                |

## 5.7 Overt versus Stealthy Prompt Injection

Table 8 compares overt and stealthy prompt-injection performance using attack-success rate, D1 catch rate and probe-dispatch rate.

*Table 8 Overt-versus-stealthy prompt-injection results*

| Baseline   | Scenario              | Style    | Episodes | Attack success | D1 catch rate | Probe dispatch |
|------------|-----------------------|----------|----------|----------------|---------------|----------------|
| No defence | Prompt injection only | Overt    | 60       | 100.0%         | N/A           | 0.0%           |
| No defence | Prompt injection only | Stealthy | 60       | 100.0%         | N/A           | 0.0%           |
| No defence | Combined              | Overt    | 60       | 100.0%         | N/A           | 0.0%           |
| No defence | Combined              | Stealthy | 60       | 100.0%         | N/A           | 0.0%           |
| D1–D3 only | Prompt injection only | Overt    | 60       | 0.0%           | 100.0%        | 0.0%           |
| D1–D3 only | Prompt injection only | Stealthy | 60       | 8.3%           | 80.0%         | 0.0%           |
| D1–D3 only | Combined              | Overt    | 60       | 0.0%           | 100.0%        | 0.0%           |
| D1–D3 only | Combined              | Stealthy | 60       | 0.0%           | 80.0%         | 0.0%           |
| Full D1–D4 | Prompt injection only | Overt    | 60       | 0.0%           | 100.0%        | 0.0%           |
| Full D1–D4 | Prompt injection only | Stealthy | 60       | 0.0%           | 80.0%         | 8.3%           |
| Full D1–D4 | Combined              | Overt    | 60       | 0.0%           | 100.0%        | 0.0%           |
| Full D1–D4 | Combined              | Stealthy | 60       | 0.0%           | 80.0%         | 0.0%           |

Figure 20 compares attack-success rates for overt and stealthy prompt-injection styles across the evaluated defence configurations.

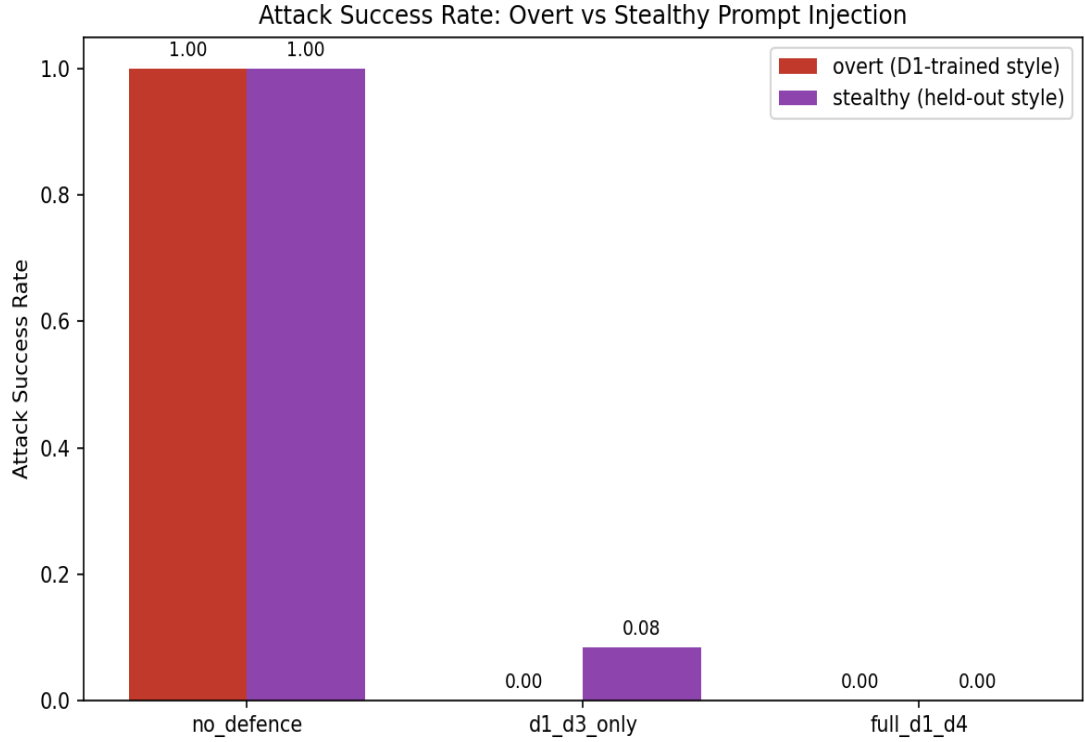


*Figure 20 Attack success rate for overt and stealthy prompt injection*

The D1 catch-rate values provide additional detail. For overt injection, D1 caught all 60 episodes in both defended configurations. For stealthy injection, D1 caught 48 of 60 episodes, giving a catch rate of 0.80. The remaining stealthy cases therefore provide evidence that later layers can become relevant when the input classifier does not recognise the attack wording.

## 5.8 Observed Contribution of D4

The clearest observed contribution of D4 was the reduction in routing-manipulation success from 0.750 to 0.683, an absolute decrease of 0.067. D4 therefore added containment in the implemented routing-manipulation condition, but it also increased intervention on clean traffic.

However, the same comparison also shows that D4 is associated with additional intervention. On clean traffic, the false positive rate increased from 0.183 with D1–D3 to 0.233 with D1–D4. For routing manipulation, held-for-review increased from 0.250 to

0.283, and probe dispatch occurred in 0.067 of episodes. These observations indicate that the active-verification layer changes both security outcomes and operational handling of uncertain cases.

## 5.9 Summary of Main Findings

The results show that D1–D3 provided strong containment for the tested prompt-related attacks, while D4 provided an additional reduction for direct routing manipulation.

These gains were accompanied by higher false-positive and held-for-review rates, demonstrating the operational cost of layered security. Table 9 summarises the principal primary-sweep results for attack success and clean-traffic intervention across the three defence configurations.

*Table 9 Summary of principal primary-sweep results*

| Measure                      | No defence | D1–D3 only | Full D1–D4 |
|------------------------------|------------|------------|------------|
| Combined attack success      | 100.0%     | 0.0%       | 0.0%       |
| Routing manipulation success | 100.0%     | 75.0%      | 68.3%      |
| Prompt injection success     | 100.0%     | 0.0%       | 0.0%       |
| Clean false positive rate    | 0.0%       | 18.3%      | 23.3%      |

## **Chapter 6: Evaluation and Discussion**

### **6.1 Evaluation Overview**

The central outcome measure was attack success, defined in the implementation as an attack being present while the final action was neither held for review nor used to isolate the affected node. Additional measures included false positive rate on clean traffic, held-for-review rate, probe-dispatch rate and D1 catch rate. These measures are useful together because a security mechanism that reduces attack success but frequently interrupts legitimate operation may introduce an operational cost.

### **6.2 Evaluation Against the Research Objectives**

Objective 1 was addressed through the critical review of FANET routing, UAV security, prompt injection, intrusion detection and recent LLM-agent research. Objective 2 was addressed through the implemented Python simulation and D1–D4 framework.

Objective 3 was addressed through controlled comparisons of attack success, false-positive rate, held-for-review rate, probe dispatch and prompt-style performance.

The three objectives defined in Chapter 1 can be assessed directly from the completed literature review, implementation and experimental results.

### **6.3 Overall Security Effectiveness**

D1–D3 prevented successful outcomes in the tested prompt-injection-only and combined scenarios, while routing-manipulation-only success remained 0.750.

### **6.4 Discussion of D1: Prompt-Injection Detection**

D1 provides the first security boundary at the prompt level. Its performance was strongly dependent on attack style. In the style-comparison experiment, overt prompt injection was caught in all 60 episodes by the defended configurations, giving a D1 catch rate of 1.00. For stealthy prompt injection, D1 caught 48 of 60 episodes, giving a catch rate of 0.80. This difference is particularly relevant because the D1 training data contained overt injection examples while stealthy wording was held out as a separate condition.

The most important observation is that the 0.80 D1 catch rate for stealthy attacks did not translate into successful attacks in the full prompt-injection-only evaluation. The full pipeline still recorded 0.000 attack success. This provides direct evidence for the value

of layering: a prompt detector can miss some malicious inputs without necessarily allowing those inputs to produce an unsafe final routing action.

## **6.5 Discussion of D2 and D3: Independent Routing and Behavioural Checks**

D2 and D3 provide security boundaries that are independent of prompt classification. D2 checks whether a candidate routing decision is consistent with the simulated network state, while D3 examines runtime indicators such as route churn, queue length, route count and battery state. This separation is consistent with UAV intrusion-detection research that uses behavioural and cyber-physical evidence rather than relying only on input inspection [9].

This finding is compatible with previous UAV security research, where intrusion detection has been used to identify cyber-physical attacks through network and behavioural features. Mughal et al. demonstrated the value of combining cyber and physical information in UAV swarm intrusion detection [9]. Similarly, Benfriha et al. investigated learning-based mitigation of routing attacks in FANETs [8]. The present implementation differs because the monitored decision chain includes an AI-generated routing recommendation and prompt-level evidence. The results therefore support combining AI-layer checks with network and behavioural monitoring rather than treating them as interchangeable mechanisms.

## **6.6 Discussion of D4: Active Verification**

D4 was introduced to address cases that remained uncertain after the preceding controls. Its most direct contribution can be seen in the routing-manipulation-only scenario, where attack success decreased from 0.750 with D1–D3 to 0.683 with D1–D4. Probe dispatch also occurred in 0.067 of routing-manipulation-only episodes, while held-for-review increased from 0.250 to 0.283. These observations indicate that D4 was not merely an additional classifier; it changed the decision process by obtaining further evidence before final handling of some suspicious cases.

The D4 results also demonstrate an important limitation. The probe was modelled with a simulated reliability rather than measured using physical UAV experiments. Consequently, the observed reduction from 0.750 to 0.683 demonstrates the behaviour of the implemented verification policy under its defined assumptions, not the expected

performance of a real UAV probe. This caveat is essential when interpreting the contribution of D4.

## **6.7 Overt versus Stealthy Attacks**

The style-comparison experiment provides an important test of generalisation within the defined attack model. Under no defence, both overt and stealthy prompt injection produced 1.000 attack success. Under D1–D3, overt prompt injection produced 0.000 attack success, whereas stealthy prompt injection produced 0.083. Under the full D1–D4 configuration, both styles produced 0.000 attack success in the prompt-injection-only scenario.

The 0.083 result under D1–D3 is particularly useful because it exposes a weakness that is hidden by testing overt attacks alone. It demonstrates that D1's 0.80 catch rate for stealthy wording leaves some cases unresolved at the input layer. The full pipeline's 0.000 attack success shows that later controls can compensate for some missed prompt-level detections. This supports the use of held-out attack styles when evaluating security mechanisms, rather than relying exclusively on attack patterns like the training data.

## **6.8 Limitations of the Evaluation**

The principal limitation is the use of a deterministic local LLM surrogate. It was selected to make repeated experiments controlled and reproducible, but it does not reproduce the full behavioural variability, contextual reasoning or failure modes of a commercial or large open-weight LLM. The results therefore establish how the framework behaves against the implemented surrogate and attack model, not against LLMs in general.

A second limitation is the use of project-generated synthetic data. The D1 detector was trained on 1,200 synthetic telemetry-log samples and the D3 detector on 500 synthetic behavioural samples. These data are useful for controlled experimentation, but they do not represent the full diversity of real UAV telemetry or real-world prompt-injection language. The stealthy condition partly addresses this issue by holding out a different prompt style, but it remains within the project's synthetic attack construction.

## **6.9 Answer to the Research Question**

The research question asked: “How effectively can a multi-layer security framework detect and reduce prompt-injection-induced routing manipulation in AI-assisted UAV

swarm networks, and to what extent can an active verification layer improve the verification and containment of suspicious routing behaviour that may bypass earlier defence mechanisms?”

Within the implemented simulation, the framework was effective against the tested prompt-injection-only and combined scenarios, where both defended configurations recorded 0.000 attack success compared with 1.000 without defence. For routing manipulation without prompt injection, the passive D1–D3 configuration reduced attack success to 0.750 and D4 reduced it further to 0.683. The active layer therefore provided an additional observed reduction of 6.7 percentage points under the tested routing-manipulation condition.

## **6.10 Chapter Summary**

This chapter critically evaluated the experimental findings against the research objectives and relevant literature. The results show that no-defence execution allowed all tested attack scenarios to succeed, while D1–D3 prevented successful outcomes in the prompt-injection-only and combined scenarios. Routing manipulation remained more difficult, with attack success falling from 0.750 under D1–D3 to 0.683 when D4 was added.

## **Chapter 7: Conclusion and Recommendations**

### **7.1 Conclusion**

This project investigated the security risks of LLM-assisted routing in autonomous UAV networks and developed a four-layer framework comprising D1 prompt-injection detection, D2 routing-output validation, D3 behavioural monitoring and D4 active verification. The Python-based simulation demonstrated that the no-defence configuration allowed all tested attacks to succeed, while D1–D3 prevented successful prompt-injection-only and combined attacks. However, routing manipulation without prompt injection remained a residual weakness, with attack success reduced from 1.000 to 0.750 by D1–D3 and to 0.683 by the full D1–D4 framework. The results also showed lower D1 detection for stealthy injection (0.80) than overt injection (1.00), although the full framework recorded 0.000 attack success for both tested styles. Overall, the study demonstrates the value of combining multiple security boundaries with active evidence gathering, while the conclusions remain limited by the deterministic surrogate, synthetic data, simplified FANET model and simulated probe reliability.

### **7.2 Recommendations**

The framework should be improved by training D1 on a broader range of direct, indirect, obfuscated and stealthy prompt injections, while D2 and D3 should incorporate richer routing, trust and network-state constraints. Security thresholds should also balance attack containment with unnecessary intervention on legitimate traffic. D4 should be triggered selectively when earlier evidence is insufficient or conflicting, using adaptive thresholds and improved trust models to reduce unnecessary probing and held-for-review.

### **7.3 Suggestions for Further Work**

Future work should evaluate the proposed framework using real AI/LLM systems rather than the deterministic surrogate used in this study. Testing the framework with real models and realistic prompt-injection scenarios would help assess its behaviour under conditions closer to real-world UAV operations. The framework could then be implemented and tested on physical UAVs or a hardware-in-the-loop swarm environment to evaluate its practical performance under real communication, mobility and network conditions.

## **7.4 Closing Statement**

The project has produced an implemented and experimentally evaluated framework within a defined simulation scope. Its main value is the integration of multiple security boundaries and the demonstration of how active verification can provide additional evidence when passive controls are insufficient.

## Chapter 8: Reflection

Completing this project improved my ability to translate a research problem into an implemented and testable security system. At the beginning, the combination of prompt injection, LLM-assisted decision-making, UAV swarm networking and routing security was challenging. Reviewing the literature helped me understand how these areas connect and identify a practical research gap.

One of the main lessons was the importance of planning before implementation. I learned that clear objectives, attack scenarios, success criteria and evaluation measures need to be defined before experiments are conducted. During development, I also had to repeatedly check that the implementation remained consistent with the research objectives.

The project strengthened my practical skills in Python programming, simulation, machine learning and experimental analysis. Working with the dynamic UAV/FANET environment also improved my understanding of how network conditions and routing behaviour can interact with security decisions. The use of controlled experiments showed me the importance of reproducibility when evaluating a security framework.

The project also changed my understanding of security evaluation. I initially focused mainly on detecting attacks, but the results showed that stronger security controls can also introduce unnecessary intervention. This taught me to consider security, reliability and operational impact together.

From an academic and personal perspective, I improved my ability to read technical literature critically, organise a large project, analyse results and recognise limitations. If I repeated the project, I would begin experimental planning earlier, allow more time for validation and test the framework under more realistic conditions.

Overall, this project has developed both my technical and research skills and has given me practical experience in progressing from literature review and system design to implementation, experimentation, analysis and reflection. It has also increased my confidence in tackling interdisciplinary security problems and prepared me for more advanced research or professional work in AI-enabled autonomous systems.

## References

- [1] O. S. Oubbati, M. Atiquzzaman, P. Lorenz, M. H. Tareque, and M. S. Hossain, "Routing in Flying Ad Hoc Networks: Survey, Constraints, and Future Challenge Perspectives," *IEEE Access*, vol. 7, pp. 81057–81105, 2019.
- [2] Y.-Y. Chen, W.-C. Huang, C.-L. Lin, S.-H. Chen, and C.-Y. Lu, "Development of Adaptive Drone Swarm Networks," *IEEE Access*, vol. 13, pp. 131582–131599, 2025.
- [3] Z. Du, C. Luo, G. Min, J. Wu, C. Luo, J. Pu, and S. Li, "A Survey on Autonomous and Intelligent Swarms of Uncrewed Aerial Vehicles (UAVs)," *IEEE Transactions on Intelligent Transportation Systems*, vol. 26, no. 10, pp. 14477–14500, 2025.
- [4] N. Zhang, C. Liu, and J. Ba, "Decomposing FANET to Counter Massive UAV Swarm Based on Reinforcement Learning," *IEEE Communications Letters*, vol. 27, no. 7, pp. 1784–1788, 2023.
- [5] A. V. Leonov, "Modeling of bio-inspired algorithms AntHocNet and BeeAdHoc for Flying Ad Hoc Networks (FANETs)," in *Proc. 13th Int. Scientific-Technical Conf. Actual Problems of Electronics Instrument Engineering (APEIE)*, vol. 2, 2016, pp. 90–99.
- [6] G. Zhu, H. Yao, T. Mai, Z. Wang, D. Wu, and S. Guo, "Fission Spectral Clustering Strategy for UAV Swarm Networks," *IEEE Transactions on Services Computing*, vol. 17, no. 2, pp. 537–548, 2024.
- [7] D.-Y. Kim and J.-W. Lee, "Topology construction for flying ad hoc networks (FANETs)," in *Proc. ICTC*, 2017, pp. 153–157.
- [8] S. Benfriha, M. Babaghayou, H. B. Salameh, and S. Benadla, "A Robust Federated Deep Learning-Based Framework to Mitigate Routing Attacks in Flying Ad Hoc Networks," in *Proc. SDS*, 2024, pp. 99–103.
- [9] U. A. Mughal, S. C. Hassler, and M. Ismail, "Machine Learning-Based Intrusion Detection for Swarm of Unmanned Aerial Vehicles," in *Proc. IEEE Conference on Communications and Network Security (CNS)*, 2023, pp. 1–9.
- [10] A. Altaweel, H. Mukkath, and I. Kamel, "GPS Spoofing Attacks in FANETs: A Systematic Literature Review," *IEEE Access*, vol. 11, pp. 55233–55280, 2023.
- [11] M. Rizk, F. Slim, and J. Charara, "Toward AI-Assisted UAV for Human Detection in Search and Rescue Missions," in *Proc. DASA*, 2021, pp. 781–786.
- [12] F. Mohamadi, "AI-Assisted UAV Swarms Detection," *2024 Conference on AI, Science, Engineering, and Technology (AIxSET)*, Laguna Hills, CA, USA, 2024, pp. 208–211.
- [13] M. A. Ferrag, A. Lakas, N. Tihanyi, and M. Debbah, "LLM and AI Agents for Autonomous Systems: A Survey of Applications, Datasets, and Security Challenges," *IEEE Open Journal of Intelligent Transportation Systems*, vol. 7, pp. 615–657, 2026.
- [14] Y. Tian, F. Lin, Y. Li, T. Zhang, Q. Zhang, X. Fu, J. Huang, X. Dai, Y. Wang, C. Tian, B. Li, Y. Lv, L. Kovács, and F.-Y. Wang, "UAVs Meet LLMs: Overviews and Perspectives Toward Agentic Low-Altitude Mobility," *Information Fusion*, vol. 122, Art. no. 103158, 2025.
- [15] J. D. Duarte, G. D. Cândido, J. R. A. De Britto Filho, J. Souza Neto, E. J. da Costa, J. P. J. da Costa, and L. Peotta de Melo, "A Systematic Review of Prompt Injection

- Attacks on Large Language Models: Trends, Taxonomy, Evaluation, Defenses, and Opportunities,” *IEEE Access*, vol. 14, pp. 12875–12899, 2026.
- [16] D. W. Yip, A. Esmradi, and C. F. Chan, “A Novel Evaluation Framework for Assessing Resilience Against Prompt Injection Attacks in Large Language Models,” in *Proc. IEEE Asia-Pacific Conference on Computer Science and Data Engineering*, 2023, pp. 1–5.
  - [17] OWASP, “LLM01:2025 Prompt Injection,” OWASP GenAI Security Project, 2025.
  - [18] A. E. Ergün and A. Onan, “Adversarial Prompt Detection in Large Language Models: A Classification-Driven Approach,” *Computers, Materials & Continua*, vol. 83, no. 3, pp. 4855–4877, 2025.
  - [19] R. Zeng, C. Zhao, Z. Liu, and Y. Lu, “UAV Anomaly Detection Model Based on Integrated Multi-Modal Neural Network and Neural Architecture Search,” in *Proc. ISPDS*, 2024, pp. 585–588.
  - [20] Q. Zeng, A. Bashir, and F. Nait-Abdesselam, “UAVIDS-2025: A Benchmark Dataset for Intrusion Detection in UAV Networks Using Machine Learning Techniques,” in *Proc. 2025 IEEE Conference on Communications and Network Security (CNS)*, 2025, pp. 1–9.
  - [21] Y. Emami, R. Taheri, M. H. Homaei, M. A. U. Rehman, and M. Shojafar, “LLM-Centric Agentic AI for UAV Swarms: Architecture, Enabling Technologies, and Open Problems,” *arXiv:2607.09756*, 2026.
  - [22] M. A. Ferrag, A. Lakas, and M. Debbah, “ $\alpha^3$ -SecBench: A Large-Scale Evaluation Suite of Security, Resilience, and Trust for LLM-based UAV Agents over 6G Networks,” *arXiv:2601.18754*, 2026.
  - [23] T. Aida, S. Hasegawa, M. Arai, R. Isogai, Y. Shoji, and M. Hasegawa, “Optimization of End-to-End Throughput on Piggyback Network with Drone-to-Drone Millimeter-Wave Communications,” in *Proc. ICOIN*, 2024, pp. 751–755.
  - [24] S. Ali, A. Kumar, and M. Alam, “Secure and Resilient Medical Payload Delivery in Disaster Zones Using Drone-to-Drone Communication and Centralized Coordination,” in *Proc. ICECI*, 2026, pp. 1–6.
  - [25] L. M. Schalk and D. Becker, “The Impact of Multipath Propagation on Cooperative Traffic Conflict Detection among Drones,” in *Proc. DASC*, 2019, pp. 1–7.
  - [26] D. Mehta, K. Patel, J. Patel, R. Gupta, S. Tanwar, A. Gupta, and I. Woungang, “Safe-Uav: An Explainable AI-assisted Framework for Securing UAV Communication Networks,” in *Proc. IEEE Globecom Workshops*, 2024, pp. 1–6.
  - [27] G. A. Kakamoukas, P. G. Sarigiannidis, and I. D. Moscholios, “High Level Drone Application Enabler: An Open Source Architecture,” in *Proc. CSNDSP*, 2020, pp. 1–4.
  - [28] O. S. Oubbati, A. Lakas, F. Zhou, M. Güneş, and M. B. Yagoubi, “A survey on position-based routing protocols for Flying Ad hoc Networks (FANETs),” *Vehicular Communications*, vol. 10, pp. 29–56, 2017.