Defensive Multi-Agent AI Systems: A Two-Layer Architecture for Adaptive Real-Time Cyber Defence

Aslan Alwi*, Munirah, Almudaya Research Institute Team
aslan.alwi@umpo.ac.id, munirah@umpo.ac.id

Independent Researcher at Almudaya Research Institute
Department of Informatics Engineering, Faculty of Engineering, Universitas Muhammadiyah Ponorogo, Indonesia

November 2025

Abstract

In this paper we introduce a novel defence paradigm tailored for the era of AI-enabled threats: a two-layer architecture called the *Defensive Multi-Agent AI System* (DMAS). The architecture comprises lightweight *Edge-Agents* deployed at sensing or attack-contact points and a centralised *Core-Agent Swarm* responsible for correlation, high-level reasoning, planning, and orchestrated responses. Edge-Agents perform local triage, context enrichment, and immediate micro-actions, while Core-Agents aggregate telemetry, employ large language models (LLMs) for advanced analysis, generate structured response playbooks, and dispatch defence instructions back to the edge. We present a benign proof-of-concept implementation that simulates telemetry collection, prompt-based request/response reasoning, and tiered mitigation logic within a sandboxed environment. Finally, we analyse the security implications of DMAS, evaluate performance metrics (MTTD, MTTR, false-positive burden), and discuss governance considerations regarding autonomous defence, explainability, and AI-safety. Our contributions include: (1) a formal model of the defensive two-layer multi-agent system, (2) a detailed architecture and communication protocol design, (3) a safe PoC implementation, and (4) a comprehensive security and governance analysis of AI-native defence systems.

1 Introduction

Cyber defence is undergoing a fundamental transformation driven by the emergence of AI-assisted and AI-orchestrated cyber threats. Recent threat intelligence reports from Google Cloud [1] and Microsoft [2] highlight that advanced threat actors, including state-sponsored groups, have begun integrating large language models (LLMs) into their operational workflows—from tool generation and reconnaissance automation to command synthesis. The SentinelOne research community has further demonstrated concrete evidence of malware frameworks embedding prompt templates and API keys to leverage external AI models dynamically [3]. These developments indicate a paradigm shift where malicious capability is no longer fully contained within a static binary but can be delegated to external AI services.

Traditional defensive tools such as EDR, XDR, SIEM, and SOAR platforms were not designed to handle threats whose behavioural patterns are dynamically generated by external AI systems. Their detection mechanisms primarily rely on signature matching, heuristic logic, or behavioural baselines, all of which are challenged by model-generated polymorphism, prompt-conditioned behavioural adaptation, and context-aware malicious actions. The increasing autonomy of offensive agents presents additional risks that extend beyond existing defensive assumptions.

To address this emerging landscape, we propose a new defence framework called the *Defensive Multi-Agent AI System* (DMAS). The approach is inspired by principles of distributed autonomous systems and multi-agent coordination long explored in security automation and cyber threat modelling [4, 5]. DMAS consists of two complementary layers:

1. **Edge-Agents**: lightweight AI-assisted agents deployed at contact points of potential attacks such as endpoints, gateways, network ingress points, and API surfaces. These agents perform fine-grained sensing, local triage, contextual enrichment, and micro-response actions under strict policy controls.

2. **Core-Agent Swarm**: a centralised cluster of coordinated, specialised AI agents responsible for deep analysis, correlation, high-level reasoning, response orchestration, and reporting. This includes an LLM-based Analyst Agent, a Planner/Orchestrator Agent, a Policy Engine, an Adaptation Agent, and a Reporting Agent.

This two-layer defence architecture rebalances intelligence between perimeter and centre. Edge-Agents provide rapid, localised response capability and continuous telemetry, while Core-Agents offer global awareness, cross-host correlation, and strategic reasoning beyond the capacity of isolated edge systems. Through structured prompt-based communication, DMAS enables AI-driven analysis and autonomous-but-governed response workflows while maintaining transparency, audibility, and human oversight.

This paper makes the following contributions:

- We introduce a formal architecture for a two-layer Defensive Multi-Agent AI System (DMAS) designed for AI-native cyber defence.

- We describe the design, communication model, policy governance, and operational workflow of both Edge-Agent and Core-Agent layers.

- We present a benign, fully sandboxed proof-of-concept (PoC) demonstrating telemetry ingestion, structured LLM-driven analysis, action recommendation, and policy-constrained automated response.

- We analyse defensive capabilities, limitations, and resilience against AI-driven threats and offensive multi-agent malware systems.

- We discuss ethical, privacy, and governance considerations for deploying partially autonomous defensive agents.

The rest of this work is organised as follows. Section III surveys existing defensive approaches, detection systems, and autonomous response frameworks. Section IV presents the threat model and defensive assumptions. Section V describes the high-level architecture of DMAS. Section VI details the Edge-Agent design, while Section VII discusses the Core-Agent Swarm. Section VIII outlines the operational lifecycle. Section IX presents a benign PoC implementation, and Section X provides a comprehensive security and governance analysis.

2 Related Works

Research in cyber defence has evolved significantly over the past decade, with major advances in endpoint detection, automated response systems, graph-based threat analysis, and more recently, AI-assisted defensive technologies. In this section, we summarise prior work relevant to the proposed Defensive Multi-Agent AI System (DMAS), highlighting the limitations of existing approaches in countering AI-driven and AI-orchestrated threats.

2.1 Traditional EDR, XDR, and SIEM-Based Defence

Endpoint Detection and Response (EDR) and Extended Detection and Response (XDR) platforms represent mainstream defensive technologies. Modern EDR systems focus on behavioural analytics, process-tree monitoring, and attack surface reduction techniques, while SIEM platforms aggregate large volumes of event logs for correlation and alerting. Products such as Microsoft Defender for Endpoint [6] and Elastic Security [7] have significantly improved the state of enterprise defence.

However, these platforms rely primarily on signature-based detection, predefined heuristics, and rule-based correlation. They are less effective against AI-enabled attacks that dynamically generate behaviour or modify their operational characteristics through LLM-driven synthesis. Static rules have limited capacity to capture polymorphic or context-aware malicious activity that evolves based on environmental input.

2.2 AI-Driven Detection and Machine Learning Approaches

Machine learning and anomaly detection techniques have been widely explored in cyber defence research. Numerous studies have demonstrated the utility of supervised and unsupervised models for intrusion detection, malware classification, and behavioural fingerprinting [8]. Cloud providers have integrated ML-based threat analytics into their security suites, such as Google Cloud's security analytics platform [9] and Microsoft Defender's AI-driven investigation graph [10].

These systems improve scalability and detection accuracy but generally lack multi-agent coordination, high-level reasoning, and dynamic orchestration capabilities. Additionally, many ML-based approaches operate on fixed feature sets and static models, which can be susceptible to adversarial attacks or become outdated when facing AI-generated, zero-shift behaviour.

2.3 Autonomous Response and SOAR Platforms

Security Orchestration, Automation, and Response (SOAR) platforms such as Palo Alto Cortex XSOAR [11] and Splunk SOAR [12] offer automated workflows for incident response. These platforms can execute predefined playbooks, aggregate signals from multiple sources, and automate containment steps.

However, SOAR systems are heavily reliant on preconfigured rules and lack adaptive reasoning and contextual decision-making. Their responses are deterministic and do not exhibit the autonomous planning or multi-agent coordination required to counter sophisticated AI-enabled adversaries. SOAR platforms do not perform multi-hop reasoning or generate strategies based on dynamic threat conditions.

2.4 Multi-Agent Systems in Cybersecurity

Prior research has examined multi-agent coordination in distributed security contexts, including swarm-based intrusion detection, cooperative defence frameworks, and distributed honeypot networks [13]. These works demonstrate that agent collaboration improves detection coverage and resilience.

Recent interest in agent-based AI tools (e.g., LangChain Agents, AutoGen-based multi-agent frameworks, and Google DeepMind's multi-agent research) highlights the feasibility of distributed, task-specialised agents. However, existing research has not explored a two-layer defensive architecture where edge and core agents coordinate through LLM-structured reasoning and policy-governed autonomous mitigation.

2.5 Gap Analysis

Across the literature, three limitations remain largely unaddressed:

1. Existing defensive systems lack adaptive reasoning comparable to AI-enabled offensive threats.
2. Current solutions do not define a structured multi-agent model with distinct layers for sensing, triage, high-level reasoning, and policy-based orchestration.
3. No defence framework provides controlled, explainable, LLM-mediated decision-making that integrates tightly with machine-governed response workflows.

These gaps motivate the development of the Defensive Multi-Agent AI System (DMAS), which aims to provide a structured, layered, and AI-native architecture for real-time cyber defence.

3 Threat Model and Assumptions

This section defines the threat model under which the proposed Defensive Multi-Agent AI System (DMAS) operates. We adopt a modern adversarial perspective aligned with observations from global threat intelligence sources such as MITRE ATT&CK [4], Microsoft Threat Intelligence [2], and Google Cloud Threat Intelligence [1]. The objective of this threat model is to characterise attackers' capabilities, delineate the system boundary, clarify trust assumptions, and identify conditions under which DMAS must remain resilient.

3.1 Threat Landscape

Adversaries increasingly leverage AI tools—especially large language models (LLMs)—to scale reconnaissance, automate malware generation, synthesise infrastructure configurations, assist in operational tradecraft, and dynamically modify malicious behaviour. Attack campaigns observed in recent analyses [3] indicate a shift toward *AI-orchestrated* and *contextually adaptive* malware that can:

- generate variant malicious code on demand,
- adjust behaviour based on host environment,
- utilise external AI endpoints for reasoning or command generation,
- evade detection by altering tactics or payloads dynamically.

Traditional EDR and SIEM systems struggle to detect such AI-enabled polymorphism, motivating the need for more adaptive defence architectures.

3.2 Adversary Capabilities

We assume the adversary may possess the following capabilities:

1. **Advanced AI Usage:** The attacker can query external LLMs, maintain prompt templates, utilise model APIs, or deploy LLM-enabled malware.
2. **Arbitrary Malware Execution:** The adversary can deploy binaries, scripts, or fileless attacks on a compromised endpoint.
3. **Command-and-Control (C2) Flexibility:** The attacker may employ both traditional C2 channels and AI-assisted channels (e.g., encoded prompts to public LLMs).
4. **Multi-Host Coordination:** The attacker may coordinate behaviour across multiple infected hosts, modify strategies based on telemetry, or deploy multi-agent malware frameworks similar to LRMAM.
5. **Adversarial Evasion:** The attacker may attempt to evade detection using:
    - prompt obfuscation,
    - behavioural perturbation,
    - adversarial examples targeting ML detectors,
    - minimal on-disk artefacts.

We do not assume that the adversary has unlimited resources, but we assume that the adversary is capable of using state-of-the-art AI-assisted techniques and can adapt strategies rapidly.

3.3 System Boundary and Trust Zones

DMAS operates across two main layers—Edge-Agents and Core Swarm Agents—each with different trust assumptions:

- **Core-Agent Swarm: Trusted Zone.** The core environment (e.g., central servers, orchestrators, LLM analysts) is assumed to be secure, authenticated, and monitored. Hardware and OS-level protections (secure boot, container isolation) are assumed to be in place.
- **Edge-Agent Layer: Semi-Trusted Zone.** Edge-Agents operate on hosts that may be exposed to compromise. They must assume potentially hostile environments, restricted resource availability, and tampering attempts.
- **External Networks: Untrusted Zone.** Public networks, external endpoints, and attacker-controlled C2 infrastructure are fully untrusted.

Communication between layers is assumed to use mutually authenticated and encrypted channels (e.g., mTLS), with message signing and replay protection.

3.4 Defensive Assumptions

DMAS assumes the following constraints for safe operation:

1. **Edge-Agent Integrity:** While an Edge-Agent may be partially compromised, we assume the agent itself is signed and validated at runtime with tamper detection.
2. **Policy-Based Autonomy:** Edge-Agents may take limited autonomous actions (Tier 0–1), but high-impact actions require approval from the Core or a human operator.
3. **LLM Governance:** All LLM-based analysis is mediated by safety controllers enforcing:
    - allowed response schemas,
    - forbidden tokens or operations,
    - no raw code execution without validation.
4. **No Implicit Trust in Host Data:** Data collected by Edge-Agents is treated as potentially manipulated or incomplete.
5. **Human Oversight:** Human operators can override Core-Agent decisions for high-severity incidents.

3.5 Adversarial Considerations and Failure Conditions

DMAS is designed to remain robust under the following adversarial conditions:

- **Edge-Agent Compromise:** If an attacker compromises a host, DMAS must still:
    - verify message integrity,

  - identify anomalous data patterns,
  - detect inconsistencies with global telemetry,
  - avoid propagation of false flags or misleading prompts.

- **Prompt Manipulation Attempts:** Attackers may attempt to force Edge-Agents to send misleading or ambiguous structured prompts. LLM analysts must be resilient to prompt injection or conflicting evidence.
- **Adversarial ML Attacks:** Threat actors may craft inputs to exploit ML models at the edge or core. DMAS must implement gradient-masking or ensemble verification to mitigate such risks.
- **Partial Connectivity Loss:** Edge-Agents must operate safely even if temporarily disconnected from the core, falling back to conservative local policies.
- **Data Poisoning Risks:** Telemetry poisoning attempts should be mitigated by integrity checks, outlier detection, and cross-host correlation from the Core-Agent Swarm.

This threat model establishes the assumptions and constraints under which DMAS is evaluated. The following sections will describe the defence architecture, operational flows, and PoC implementation.

4 System Overview: Defensive Multi-Agent Architecture

To address emerging AI-enabled cyber threats, we propose a defence architecture called the *Defensive Multi-Agent AI System* (DMAS), designed around the principles of distributed sensing, centralised reasoning, and policy-governed autonomous mitigation. DMAS consists of two cooperative layers: (1) Edge-Agents operating at points of attack contact, and (2) a Core-Agent Swarm responsible for strategic analysis, correlation, and orchestrated response. This layered structure is influenced by established defence-in-depth principles [5] and modern multi-agent system research in cybersecurity [13].

4.1 Design Goals

DMAS is designed to ensure real-time, adaptive, and explainable defence against AI-enabled adversaries. Its primary goals are:

1. **Rapid Local Detection:** provide immediate sensing and triage at the host or network edge.
2. **Centralised Reasoning:** perform high-level, cross-environment analysis using coordinated AI agents.
3. **Policy-Governed Autonomy:** allow limited autonomous actions while enforcing global safety rules.
4. **Explainability:** require LLM-based analysis to produce human-readable, structured justifications.

5. **Resilience:** maintain safe operation under adversarial conditions such as compromised hosts or partial data poisoning.

These goals ensure that DMAS can operate effectively in complex, high-volume environments where manual investigation and traditional rule-based systems may fail.

4.2 Two-Layer Architecture Overview

4.2.1 Edge-Agent Layer (Distributed Local Intelligence)

Edge-Agents are lightweight autonomous components deployed on endpoints, network gateways, cloud ingress points, API surfaces, or other potential attack contact points. They serve as localised intelligence units capable of:

- collecting real-time telemetry (process behaviour, network events, API activity),
- performing local anomaly scoring or rule-based triage,
- enriching events with contextual metadata,
- constructing structured prompts for higher-level analysis,
- executing safe micro-actions under predefined policy rules.

Because Edge-Agents operate closest to adversarial activity, they provide low-latency, granular visibility. They can also act immediately to contain low-severity or clear-cut threats without involving the central system.

4.2.2 Core-Agent Swarm Layer (Centralised High-Level Reasoning)

The Core-Agent Swarm is a cluster of specialised AI-driven agents that collectively perform strategic reasoning, long-horizon planning, and cross-host correlation. It typically includes:

- **Aggregator Agent:** ingests, normalises, and enriches events from Edge-Agents.
- **LLM Analyst Agent:** performs structured analysis, produces hypotheses, and generates reasoning summaries.
- **Planner/Orchestrator Agent:** composes multi-step mitigation plans.
- **Policy Decision Agent:** enforces global rules and approves/denies automated actions.
- **Adaptation Agent:** updates signatures, thresholds, scoring models, and detection logic in response to new threat patterns.
- **Reporting Agent:** creates human-readable incident summaries, timelines, and dashboards.

This layered separation enables a clean division between localised sensing and strategic intelligence. The Core-Agent Swarm has a global view of activities across environments and can detect coordinated, distributed, or slow-moving AI-assisted attack campaigns that Edge-Agents alone cannot identify.

4.3 High-Level Communication Model

Edge-Agents communicate with the Core-Agent Swarm using authenticated and encrypted channels (e.g., mTLS). They transmit:

- structured telemetry events,
- summarised behavioural signals,
- context-enhanced prompts describing observed anomalies.

The Core returns:

- structured reasoning outputs (in JSON schema),
- recommended local actions,
- multi-step mitigation playbooks,
- policy updates or threshold adjustments.

This structured information exchange ensures transparency and enables automation while preventing unsafe free-text behaviour.

4.4 Key Advantages of the DMAS Architecture

DMAS offers several advantages over traditional defence systems:

1. **Adaptive Defence Against AI-Enabled Attacks:** DMAS is designed to counter adversaries that utilise LLMs, automated planning, and dynamic malware generation [1].
2. **Distributed Sensing + Central Intelligence:** Edge-Agents provide micro-level visibility, while Core-Agents provide macro-level correlation and reasoning.
3. **Policy-Governed Autonomy:** DMAS prevents unsafe autonomous behaviour by filtering decisions through a central policy agent [5].
4. **Explainability and Auditability:** The LLM Analyst Agent generates structured, human-readable reasoning, consistent with requirements for transparent security operations.
5. **Resilience to Host Compromise:** DMAS treats Edge-Agent data as untrusted input and uses cross-host correlation to detect inconsistencies and poisoning attempts.

This architecture establishes the foundation on which the subsequent sections describe the detailed functions of Edge-Agents (Section VI), Core-Agent Swarm components (Section VII), the operational lifecycle (Section VIII), and a safe proof-of-concept implementation (Section IX).

5 Edge-Agent Layer: Local Contact-Point Intelligence

The Edge-Agent Layer represents the distributed sensing and initial-response component of the Defensive Multi-Agent AI System (DMAS). Edge-Agents are lightweight, partially autonomous units deployed at strategic points such as endpoints, API surfaces, network ingress gateways, cloud workloads, and edge computing nodes. Operating at the closest proximity to malicious activity, they provide low-latency detection, context gathering, and limited micro-response capabilities under strict policy governance.

This section describes the architecture, operational responsibilities, and security constraints of Edge-Agents, which form the foundational data and signal layer for higher-order reasoning performed by the Core-Agent Swarm.

5.1 Design Objectives of Edge-Agents

Edge-Agents are designed with four main objectives:

1. **Real-Time Sensing:** provide continuous, low-latency monitoring of events occurring at the point of potential attack.
2. **Contextual Enrichment:** attach relevant but safe metadata to enhance downstream reasoning accuracy.
3. **Safe Local Autonomy:** execute Tier 0–1 actions (e.g., blocking an IP, throttling an API key) without requiring central approval, following policy rules defined by the Core.
4. **Secure Prompt Construction:** assemble structured, sanitised, and authenticated prompts for consumption by Core-Agent LLM-based analysis.

These properties allow Edge-Agents to act as distributed intelligence nodes while delegating strategic decision-making to the central swarm.

5.2 Internal Architecture of an Edge-Agent

Each Edge-Agent consists of the following modular components:

- **Telemetry Collector:** gathers events such as process behaviours, file metadata, network flows, API usage logs, and runtime anomalies, consistent with modern EDR telemetry models [6, 7].
- **Local Triage Engine:** applies lightweight ML classifiers, anomaly scoring algorithms, threshold rules, and heuristics to assess potential maliciousness of an event.
- **Context Enrichment Module:** adds metadata including host configuration, recent process ancestry, API invocation history, container labels, and execution context while preserving privacy.
- **Prompt Builder:** constructs structured prompts in a machine-readable JSON schema, allowing Core-Agent LLM analysis to reason about events without requiring raw logs.

- **Policy-Governed Micro-Action Engine:** executes limited, reversible actions such as IP blocking, container pausing, or API rate limiting, under constraints pushed from the Core.
- **Integrity and Tamper Detection:** verifies the authenticity of its own binary or container using signatures and runtime integrity scans; reports possible tampering to the Core.

5.3 Telemetry Collection and Local Analysis

Edge-Agents collect multi-source telemetry similar to modern EDR systems but adapted for AI-specific threat indicators. Telemetry may include:

- process execution traces and parent-child relationships,
- unusual LLM API call patterns (e.g., repeated external prompt submissions),
- anomalous network flows to suspicious AI model endpoints [1],
- file write events involving encoded or obfuscated prompt templates [3],
- user behaviour anomalies (e.g., deviation from typical usage).

The Local Triage Engine assigns a severity score based on:

- anomaly detectors (e.g., autoencoders or clustering methods),
- behaviour-based heuristics inspired by ATT&CK techniques [4],
- event frequency and recency,
- context similarity to known benign or malicious patterns.

Only summarised telemetry and safe metadata are forwarded to the Core to minimise bandwidth and reduce privacy exposure.

5.4 Privacy Filtering and PII Minimisation

Edge-Agents enforce privacy policies before sending data to the Core. This includes:

- scrubbing usernames, URLs, payload bodies, personal identifiers,
- hashing sensitive identifiers with salt,
- summarising logs into statistical or categorical descriptors,
- removing raw text inputs to LLMs or user-provided content,
- encrypting sensitive fields with the Core's public key.

These practices follow principles from the NIST Privacy Framework [14].

5.5 Structured Prompt Construction

Instead of transmitting raw logs or free-form descriptions, Edge-Agents construct machine-readable prompt structures to ensure safe and predictable LLM processing. A typical prompt contains:

- a high-level summary of the observed anomaly,
- local triage score and triggered signals,
- enriched context metadata (scrubbed),
- a list of hypotheses inferred locally,
- constraints for the LLM Analyst Agent (e.g., JSON output only, token limits, forbidden operations).

This enables structured, explainable reasoning at the Core and reduces the risk of prompt injection or ambiguous interpretation.

5.6 Policy-Governed Local Response

The Edge-Agent can perform low-impact actions when local confidence is high and operational risk is minimal. These include:

- blocking outbound connections to high-risk domains,
- temporarily revoking API keys showing anomalous behaviour,
- pausing suspicious containers or processes,
- reducing service rate limits for anomalous clients,
- raising local alerts to administrators.

All such actions are restricted by policies issued by the Core-Agent Swarm and may require justification or approval for escalation. High-impact actions, such as full host isolation, require Core or human approval.

5.7 Resilience and Tamper Resistance

Edge-Agents must remain functional under adversarial pressure. DMAS assumes:

- the Edge-Agent binary or container is signed and verified at runtime,
- telemetry is buffered when offline and synchronised upon reconnection,
- event integrity is enforced using message signing,
- suspicious behaviour of the Edge-Agent itself is reported as a meta-event to the Core.

Even if an Edge-Agent operates on a compromised host, cross-host correlation in the Core can still identify inconsistencies and potential tampering.

This robust local layer enables DMAS to detect, triage, and respond to AI-enabled threats while maintaining privacy guarantees and controlled autonomy.

6 Core-Agent Swarm Layer: Central Reasoning and Orchestration

The Core-Agent Swarm Layer forms the analytical and decision-making backbone of the Defensive Multi-Agent AI System (DMAS). While Edge-Agents provide distributed sensing and rapid local response, the Core-Agent Swarm performs global correlation, structured reasoning, high-level planning, and policy enforcement. This layered approach aligns with the principles of zero-trust architecture [15] and the increasing need for AI-assisted security analytics observed across modern defence platforms [9, 10].

The Core-Agent Swarm consists of a set of cooperating AI-driven agents, each specialised for different aspects of defensive intelligence. Their collective function is to transform raw telemetry and local triage signals into actionable, safe, and policy-governed response workflows.

6.1 Core-Agent Roles and Responsibilities

6.1.1 Aggregator Agent

The Aggregator Agent ingests telemetry and prompt data from distributed Edge-Agents. It normalises the data, performs enrichment using external threat intelligence (e.g., MITRE ATT&CK mappings [4]), and clusters related events to identify potential campaign-level correlation. Key functions include:

- log and event normalisation (AVRO/JSON schema alignment),
- deduplication and anomaly pre-filtering,
- enrichment from intelligence feeds (IP reputation, malware families, LLM misuse indicators),
- construction of event graphs linking hosts, identities, API keys, and network entities.

6.1.2 LLM Analyst Agent

The LLM Analyst Agent performs structured reasoning on enriched events. It receives a strictly formatted prompt from the Aggregator and outputs a JSON-structured analysis, including:

- hypotheses with confidence scores,
- classification of behaviour (benign/suspicious/malicious),
- explanation of findings,
- recommended micro- and macro-response actions,
- queries for additional evidence collection.

To ensure safety, the LLM Analyst Agent is bound by:

- schema-constrained output (JSON only),
- maximum token limits,
- forbidden-token filters,
- sandboxed inference environments,
- runtime output validation.

Such constraints reflect emerging best practices in AI safety and misuse prevention [16].

6.1.3 Planner and Orchestrator Agent

The Planner/Orchestrator Agent transforms analysis data into multi-step response plans. Unlike traditional SOAR playbooks [11], the Planner uses a combination of rule-based logic, statistical models, and LLM-assisted sequencing to:

- prioritise actions across multiple hosts,
- coordinate containment and investigation,
- determine whether isolated micro-actions require escalation,
- ensure that actions comply with global policy constraints.

This agent allows DMAS to respond coherently to distributed or coordinated AI-assisted attacks.

6.1.4 Policy Decision Agent

The Policy Decision Agent acts as a governance and safety gateway. It evaluates proposed actions based on:

- enterprise security policies,
- regulatory requirements (e.g., data protection constraints),
- operational risk scores,
- asset criticality,
- human-authorisation requirements.

Actions such as full host isolation or credential revocation require approval from this agent or a human operator. This prevents overreach and ensures explainable AI-assisted decision-making.

6.1.5 Adaptation Agent

The Adaptation Agent continuously monitors system performance, false positives, telemetry patterns, and adversarial signals. It updates:

- threshold scores,
- anomaly detection models,
- prompt templates,
- feature weights for triage engines,
- policy rules.

This ensures DMAS remains resilient against evolving attacker tactics, including prompt manipulation, model drift, and AI-driven evasion attempts.

6.1.6 Reporting and Audit Agent

The Reporting Agent generates human-readable artefacts, including:

- incident timelines,
- analyst-friendly summaries,
- root-cause explanations,
- compliance-ready audit logs,
- dashboards for SOC operators.

It ensures transparency and supports human oversight, a key requirement for AI governance frameworks [5].

6.2 Distributed Reasoning Pipeline

The Core-Agent Swarm operates in a multi-stage pipeline:

1. **Ingest:** events, telemetry, and prompts arrive from the Edge-Agent Layer.
2. **Enrichment:** Aggregator Agent normalises and enriches data.
3. **Reason:** LLM Analyst Agent performs structured analysis on enriched events.
4. **Plan:** Planner/Orchestrator creates a multi-step mitigation or investigation workflow.
5. **Decide:** Policy Agent approves or rejects proposed actions.
6. **Enforce:** Approved actions are sent to Edge-Agents.
7. **Explain:** Reporting Agent produces summaries for humans.

This modular approach mirrors distributed multi-agent architectures examined in prior cybersecurity models [13] but introduces structured LLM reasoning and policy-governed autonomy, which were previously unexplored.

6.3 LLM Governance and Output Safety

AI-driven defence must adhere to strict safety guidelines to prevent misuse or accidental harm. Therefore, DMAS enforces:

- strict schema validation of LLM outputs,
- deny-lists forbidding unsafe commands or privileged instructions,
- time-limited and audit-logged inference sessions,
- human review for high-severity or high-impact actions.

LLM prompts themselves are constructed by Edge-Agents using privacy filters, minimising exposure of sensitive information.

6.4 Scalability and Performance Considerations

The Core-Agent Swarm is designed for large-scale deployment across enterprise and cloud environments. It employs:

- horizontal scaling for ingestion and aggregation,
- batching and prioritisation for high-volume events,
- asynchronous task queues for long-horizon reasoning,
- distributed caching for event correlation,
- GPU-backed LLM inference clusters for rapid turnaround.

These design choices align with scalable cloud security analytics architectures such as those in Google Cloud Security [9] and Microsoft Sentinel [17].

This centralised intelligence layer enables DMAS to analyse, coordinate, and orchestrate defence actions across complex, distributed environments, providing resilience against emerging AI-enabled threats.

7 Operational Lifecycle of the Defensive Multi-Agent AI System

The Defensive Multi-Agent AI System (DMAS) operates as a coordinated, multi-stage defence pipeline that spans from distributed sensing at the edge to high-level reasoning and policy-enforced response in the core. The lifecycle is designed to provide timely, adaptive, and explainable defensive actions against AI-enabled and dynamically evolving cyber threats. This section describes the end-to-end workflow, including sequence logic, state transitions, and fallback strategies for degraded or adversarial conditions.

7.1 Overview of the Lifecycle

The DMAS lifecycle consists of ten major stages:

1. **Event Sensing (Edge)** The Edge-Agent observes anomalous or noteworthy system behaviours via telemetry collection modules. Telemetry sources include process activity, network flows, file-system events, identity and credential use, and AI-related signals such as outbound LLM API calls [1, 3].

2. **Local Triage and Scoring (Edge)** Lightweight anomaly detectors, rule-based signatures, and heuristics derived from frameworks such as MITRE ATT&CK [4] generate a local triage score. This score determines whether the event is forwarded, suppressed, or acted upon locally.

3. **Context Enrichment (Edge)** Metadata such as host configuration, process ancestry, user context, container labels, and previous event history are appended to provide structured situational awareness while applying privacy safeguards defined by NIST Privacy Framework guidelines [14].

4. **Structured Prompt Construction (Edge)** The Edge-Agent composes a machine-readable prompt describing the event, the triage score, and contextual metadata. The prompt follows a strict schema and sanitisation rules to ensure safe LLM reasoning and prevent injection attacks [16].

5. **Secure Transmission to the Core** The structured prompt is authenticated and transmitted to the Core-Agent Swarm using mutually authenticated, encrypted channels (e.g. mTLS). Message integrity is ensured using cryptographic signatures.

6. **Event Ingestion and Enrichment (Core)** Upon arrival at the core, the Aggregator Agent normalises the prompt, enriches it with external threat intelligence feeds (e.g., IP reputation, AI-abuse indicators), and correlates it with other cross-host events.

7. **LLM-Driven Analysis (Core)** The LLM Analyst Agent receives the enriched event and performs structured reasoning. It outputs hypotheses, confidence scores, recommended containment steps, and additional queries, strictly within a JSON schema validated by safety controllers to enforce AI governance [10].

8. **Planning and Policy Decision (Core)** The Orchestrator Agent composes a multi-step mitigation plan. The Policy Decision Agent evaluates proposed actions against enterprise policy, risk level, regulatory constraints, and required human approvals. High-impact actions (e.g., host isolation) may require multi-factor human confirmation.

9. **Enforcement and Edge Execution** Approved actions are transmitted back to the Edge-Agent. The Policy-Governed Micro-Action Engine executes allowed actions autonomously. Actions outside its scope (Tier 2 and above) require Core or human authorisation.

10. **Feedback Loop and Adaptation** Edge-Agents report results of actions and follow-up telemetry. The Adaptation Agent updates detection thresholds, model weights, and policy parameters based on system performance, false positives, and adversarial indicators.

This closed-loop design enables continuous monitoring, reasoning, and adaptive defence in real time.

7.2 Lifecycle Sequence Diagram (Textual)

The following describes the sequence of interactions:

```
Edge-Agent:
    1. Collect telemetry
    2. Compute local triage score
    3. Build structured prompt

Core Aggregator:
    4. Receive prompt
    5. Normalise & enrich
    6. Forward to LLM Analyst

LLM Analyst Agent:
    7. Perform structured reasoning
    8. Return JSON recommendations

Planner/Orchestrator:
    9. Build multi-step plan

Policy Decision Agent:
    10. Approve/reject/modify plan

Core -> Edge:
    11. Send approved actions

Edge-Agent:
    12. Execute local micro-actions
    13. Send feedback to core

Core Adaptation Agent:
    14. Update thresholds/models
```

This deterministic pipeline ensures transparency and auditability.

7.3 Failure Modes and Fallback Strategies

DMAS is designed to operate safely under adversarial or degraded conditions. Key failure modes and mitigation strategies include:

7.3.1 1. Edge-Agent Host Compromise

If the host is compromised, the Core-Agent Swarm detects:

- inconsistent telemetry patterns,
- missing or malformed prompts,
- deviations from expected behavioural baselines.

The Core may trigger heightened monitoring or remote quarantine.

7.3.2 2. Prompt Injection or Manipulation Attempts

LLM Analyst Agents enforce:

- schema-validation,
- token blacklists,
- sandboxed inference operations,
- meta-analysis signals to detect contradictory evidence [16].

7.3.3 3. Network Loss Between Edge and Core

Edge-Agents fall back to:

- conservative local policies,
- buffered telemetry,
- no execution of high-impact actions until reconnection.

7.3.4 4. Data Poisoning or Adversarial Telemetry

The Core detects poisoning attempts using:

- cross-host correlation,
- inconsistency checking,
- anomaly clusters.

7.3.5 5. LLM Reasoning Failure

If the LLM Analyst fails validation, the system:

- discards the output,
- falls back to a rules-based contingency plan,
- alerts a human operator.

7.4 Security and Governance Integration

The operational lifecycle reflects best practices from NIST, MITRE, and major security vendors. The design ensures:

- explainability at each reasoning step,
- full audit trails,
- safe, policy-governed autonomy,
- strict separation of sensing, reasoning, and enforcement,
- resilience against AI-enabled threats.

This lifecycle provides a unified view of how DMAS coordinates distributed agents to deliver real-time adaptive defence.

8 Proof-of-Concept (PoC) Implementation

To evaluate the feasibility of the Defensive Multi-Agent AI System (DMAS), we developed a benign and fully sandboxed proof-of-concept (PoC) implementation. The PoC models both the Edge-Agent and Core-Agent Swarm interactions in a controlled environment and demonstrates structured prompting, JSON-constrained LLM reasoning, safe policy-based action enforcement, and adaptive feedback. No real malware, system manipulation, or intrusive operations are performed.

8.1 PoC Objectives

The PoC aims to demonstrate:

1. structured telemetry processing at the Edge-Agent,
2. construction of safe and minimal prompts,
3. secure event transmission to the Core,
4. LLM-driven analysis under strict schema constraints,
5. policy-based response generation,
6. safe execution of local micro-actions,
7. a complete closed-loop feedback cycle.

All code executed in the PoC is intentionally limited to simulated events and mock actions, following established best practices for responsible cybersecurity experimentation [18].

8.2 PoC Architecture

The PoC implementation consists of two main components:

- **Edge-Agent PoC**, implemented in Python, simulating telemetry collection and prompt generation.
- **Core-Agent PoC**, consisting of:
  - a mock Aggregator,
  - a schema-constrained LLM Analyst (simulated),
  - a rule-based Policy Decision Engine,
  - a mock Orchestrator for action sequencing.

No real external LLM endpoints or system commands are used. Instead, responses are generated using deterministic templates to emulate safe LLM behaviour.

8.3 Edge-Agent PoC

The Edge-Agent PoC runs a simulated event pipeline:

1. Generates a synthetic telemetry event: CPU spike, suspicious network connection, or abnormal process tree.
2. Applies a simple rule-based triage:

```
if "unknown_process" in event:
     score = 0.82
else:
     score = 0.15
```

3. Enriches with safe metadata (no PII): hostname, container ID, event timestamp.
4. Constructs a machine-readable JSON prompt:

```
{
  "event_type": "...",
  "triage_score": 0.82,
  "context": { ... },
  "constraints": {
    "output_format": "JSON",
    "max_tokens": 350,
    "forbidden_ops": ["delete","shell","network"]
  }
}
```

The Edge-Agent then sends this prompt to the Core using HTTPS simulation (without real network operations).

8.4 Core-Agent PoC

The Core-Agent PoC implements a simplified reasoning pipeline:

1. **Aggregation**: Normalises the incoming event and checks for schema validity.
2. **Mock LLM Analyst**: Generates a static JSON reasoning output based on triage score:

```
{
  "hypotheses": [
     {"id":"h1","desc":"Potential anomaly","confidence":0.71}
  ],
  "recommended_actions":[
     {"id":"a1","type":"throttle","target":"container","priority":1
  ],
  "additional_queries":[
     "collect_recent_logs"
  ]
}
```

   This simulates a structured LLM reasoning output without invoking any real inference model.

3. **Planning**: The Orchestrator sequences the recommended actions and checks policy constraints.
4. **Policy Evaluation**: A rule-based engine approves or denies actions based on:
   - severity,
   - risk,
   - autonomy tier,
   - verification of safe action types.

The Core then returns a safe and minimal action set to the Edge.

8.5 Action Execution and Policy Constraints

To ensure safety, actions performed by the Edge-Agent PoC are limited to mock operations, such as printing:

```
[Action] Would throttle container: xyz123
```

No system calls, no process manipulation, and no network interference occur. This adheres to safe-testing guidance from NIST [18].

8.6 Illustrative Event-to-Response Example

A synthetic process anomaly triggers the following sequence:

1. Edge-Agent: detects unusual process `unknown_proc`.
2. Triage engine: assigns score 0.82.
3. Prompt builder: constructs structured prompt.
4. Core Aggregator: normalises input.
5. LLM Analyst: returns JSON with hypothesis and action.
6. Orchestrator: sequences `throttle-container`.
7. Policy engine: approves Tier 1 action.
8. Edge-Agent: executes mock action.
9. Adaptation Agent: updates triage rule weights.

This demonstrates the full defensive loop of DMAS in a safe environment.

8.7 Safety Constraints in the PoC

The PoC enforces:

- no raw logs containing sensitive data,
- no external API calls,
- no system-level operations,
- no network scanning or interference,
- strict JSON schemas for mock LLM output,
- deterministic and non-destructive actions.

These restrictions align with responsible-disclosure and safe-experiment standards for cyber-security research.

8.8 Code Availability

The PoC code is suitable for inclusion in a public pre-print GitHub repository. All components operate on synthetic data and do not perform real intrusions or system alterations. The code is intentionally simple and minimal, allowing researchers to reproduce the conceptual flow without risk.

In Section 10, we evaluate the security and governance implications of the DMAS architecture and discuss its operational robustness.

9 Security Analysis

This section evaluates the security properties of the Defensive Multi-Agent AI System (DMAS), including its strengths, limitations, adversarial robustness, and governance considerations. The analysis draws upon established cybersecurity frameworks such as MITRE ATT&CK [4], NIST Zero Trust Architecture [15], and emerging research on AI-enabled threats [1, 2].

9.1 Security Strengths

DMAS introduces several advantages for defending against AI-driven, adaptive, and distributed threats.

9.1.1 1. Distributed Sensing with Centralised Intelligence

Traditional EDR/XDR systems often suffer from local blind spots and limited correlation capabilities. DMAS combines:

- **fine-grained sensing** from Edge-Agents,
- **cross-host correlation** from the Core,
- **LLM-powered reasoning** for contextual decision-making.

This hybrid distribution enables the detection of low-signal, distributed, or multi-stage attacks that could otherwise evade host-level detection.

9.1.2 2. Structured LLM Reasoning with Safety Controls

LLM-based analysis is restricted by:

- strict JSON schemas,
- token limits,
- forbidden-operation filters,
- sandboxed execution environments,
- runtime validation of outputs.

This mitigates risks of LLM misuse, output hallucination, or unbounded autonomy, aligning with AI security guidelines from Google Cloud [16].

9.1.3 3. Policy-Governed Autonomy

Unlike autonomous defence models that operate without oversight, DMAS ensures that:

- high-impact actions require human approval,
- the Policy Decision Agent enforces enterprise-wide rules,
- audit logs record every decision and reasoning step.

This supports compliance with Zero Trust principles [15] and enterprise governance.

9.1.4 4. Resilience to Host Compromise

Because Edge-Agent telemetry is treated as semi-trusted:

- inconsistencies are detected via cross-host reasoning,
- anomaly clusters expose telemetry poisoning attempts,
- tampered Edge-Agents generate meta-alerts.

This allows recovery even when a host is under adversarial control.

9.1.5 5. Adaptation Against Evolving Threats

The Adaptation Agent updates thresholds, detection rules, and prompt templates in response to:

- changes in attacker behaviour,
- false positives or false negatives,
- model drift,
- new indicators of AI-enabled abuse [3].

This mitigates stagnation and improves long-term robustness.

9.2 Security Limitations

Despite its strengths, DMAS has inherent constraints.

9.2.1 1. Partial Dependence on LLM Reasoning

Although schema-constrained, LLM reasoning still carries risk of:

- incomplete or ambiguous conclusions,
- sensitivity to adversarial prompt manipulation,
- model drift or domain misalignment.

Human oversight is required for high-impact decisions.

9.2.2 2. Edge-Agent Exposure to Host-Level Attacks

Edge-Agents operate in potentially hostile environments. Attackers may:

- intercept or manipulate telemetry,
- attempt to tamper with agent binaries,
- block transmission channels,
- mimic benign processes to evade local triage.

Tamper-resistance and cross-host correlation mitigate but cannot fully eliminate these risks.

9.2.3 3. Core-Agent Swarm as a High-Value Target

The Core systems hold global situational awareness and policy authority. Adversaries may attempt:

- API-level misuse,
- DoS attacks against aggregation or LLM inference,
- poisoning through crafted synthetic telemetry,
- privilege escalation within the orchestration pipeline.

Defensive hardening, rate limiting, and attested execution environments are essential.

9.2.4 4. Latency Constraints

Real-time defence requires:

- fast LLM inference,
- low-latency transmission,
- minimal prompt complexity.

In extremely high-volume environments, LLM inference bottlenecks may delay decision-making.

9.2.5 5. Detection Blind Spots for Novel AI Behaviours

Novel, zero-shift AI-driven behaviours may avoid:

- existing anomaly detectors,
- prompt templates,
- heuristics inherited from classical EDR models.

This requires continuous model adaptation.

9.3 Adversarial Robustness Analysis

DMAS incorporates defences against a variety of adversarial techniques:

9.3.1 1. Prompt Injection and Manipulation

LLM Analyst Agents enforce:

- sanitised input prompts,
- output schema enforcement,
- adversarial string detection.

9.3.2 2. Data Poisoning

The Core-Agent Swarm validates incoming telemetry via:

- cross-host consistency checks,
- clustering of abnormal event patterns,
- signature validation and integrity checks.

9.3.3 3. Adversarial Examples Against Models

Edge models use:

- simplified, interpretable ML models (less vulnerable to gradient attacks),
- ensemble scoring,
- fallback to rules-based triage when confidence is low.

9.3.4 4. Telemetry Mimicry by AI-Driven Malware

LLM-generated malware may attempt to mimic benign patterns [1]. DMAS counters this with:

- behaviour-time correlation,
- frequency analysis,
- temporal graph embeddings across hosts.

9.4 Failure and Abuse Scenarios

We consider potential failure scenarios:

- **LLM hallucination**: mitigated via schema validation and safety controllers.
- **Policy misconfiguration**: mitigated via audit trails and human oversight.
- **Edge-Agent isolation**: mitigated via conservative local fallback policies.
- **Orchestrator overreach**: mitigated via Policy Decision Agent's approval gate.
- **Model drift**: mitigated via continuous adaptation procedures.

9.5 Governance, Transparency, and Compliance

DMAS adheres to principles of:

- least privilege,
- explainable AI,
- minimisation of personal data,
- clear human-in-the-loop responsibilities,
- full auditability of decisions.

These requirements align with global efforts around trustworthy and secure AI deployment [19].

10 Ethics, Privacy, and Governance

The deployment of AI-assisted defensive systems introduces significant ethical, privacy, and governance considerations. As DMAS integrates automated decision-making, distributed telemetry gathering, and LLM-based analysis, it must adhere to principles of responsible AI, data minimisation, transparency, explainability, and human oversight. This section examines these issues in alignment with established guidance from NIST AI RMF [19], the NIST Privacy Framework [14], and OECD AI principles [20].

10.1 Ethical Principles for AI-Driven Cyber Defence

DMAS is built upon the following ethical principles:

- **Beneficence:** The system aims to protect assets and users while avoiding unnecessary interference with normal system operations.
- **Non-maleficence:** Automated defensive actions must not cause harm such as data loss, service disruption, or over-enforcement. All high-impact decisions require human approval.

- **Proportionality:** Responses must be proportional to the risk level. Tiered autonomy ensures that Edge-Agents only execute safe, reversible actions.
- **Justice and Fairness:** Defensive decisions must avoid bias, including unfair targeting of users, applications, or infrastructure based on incomplete or skewed telemetry.
- **Explainability:** LLM Analyst Agents produce structured justifications to support trustworthiness and auditability.
- **Accountability:** All actions taken by automated agents are logged. Operators remain the ultimate authority for high-severity interventions.

These principles reflect broader AI governance frameworks and support responsible adoption of autonomous cybersecurity systems.

10.2 Privacy Considerations

The distributed nature of DMAS introduces privacy risks related to telemetry collection and cross-host event correlation. To mitigate these risks, DMAS employs several privacy-preserving mechanisms aligned with the NIST Privacy Framework [14]:

- **Data Minimisation:** Edge-Agents scrub or hash sensitive fields before transmission. Telemetry is summarised rather than recorded verbatim.
- **Purpose Specification:** Collected data is used strictly for defensive purposes. DMAS enforces strict role-based access controls (RBAC).
- **Local Data Processing:** Context enrichment is performed on-device to minimise remote exposure.
- **Selective Disclosure:** Only essential metadata (e.g., event type, severity score, host state) is sent to the Core-Agent Swarm.
- **Encryption:** Sensitive values are encrypted at the edge using the Core's public key.
- **Telemetry Lifespan Limitation:** Raw telemetry is retained only as long as necessary for incident analysis, consistent with modern privacy guidelines.

10.3 Governance and Human Oversight

DMAS incorporates multi-level governance models to maintain operational and ethical integrity:

- **Human-in-the-Loop:** All Tier 2 and Tier 3 actions (e.g., host isolation, credential revocation) require human approval, supporting accountable autonomy.
- **Policy Decision Agent:** Acts as a governance gate ensuring that proposed actions comply with organisational policies, regulatory constraints, and operational risk scores.

- **Audit Logging:** Every decision, justification, and action is logged for forensic analysis and compliance.
- **Separation of Duties:** Different DMAS components (Edge, Core, Policy, Reporting) operate under distinct privileges to reduce the chance of systemic failure.
- **Compliance Alignment:** The system aligns with compliance requirements such as GDPR, HIPAA, and regional cybersecurity regulations by enforcing data minimisation and human oversight.

10.4 Risks of Autonomous Defence Systems

Despite the safeguards, AI-driven cyber defence introduces risks:

- **Over-Enforcement Risk:** Incorrect triage or LLM misinterpretation may lead to aggressive responses. Policy gates and human review reduce this risk.
- **Bias and Unfair Targeting:** Model drift or skewed data may cause systematic bias. The Adaptation Agent and audit reviews are responsible for bias mitigation.
- **Privacy Leakage:** If misconfigured, telemetry could reveal sensitive data. Privacy filters address this risk directly.
- **Dependency on LLM Reasoning:** Over-reliance on LLM outputs could propagate errors or hallucinations. Strict output validation and fallback logic mitigate this issue.
- **Adversarial Misuse:** Attackers may attempt to manipulate inputs to influence LLM decisions [16]. Schema validation and tamper detection form the first line of defence.

10.5 Ethical Research and Safe Experimentation

The PoC implementation is designed in accordance with NIST guidance on safe cybersecurity testing [18]. It uses only:

- synthetic data,
- mock events,
- simulated prompts,
- non-destructive action models,
- isolated, non-networked execution environments.

This ensures that DMAS research does not involve harmful code, real malware, or operational disruption.

Overall, DMAS is built as a privacy-conscious, ethically governed, and explainable defensive framework suitable for modern AI-enabled cybersecurity environments.

11 Conclusion

This paper introduced the Defensive Multi-Agent AI System (DMAS), a two-layer defence architecture designed to address the emerging class of AI-enabled cyber threats. By combining lightweight, distributed Edge-Agents with a centralised Core-Agent Swarm that leverages LLM-assisted reasoning, DMAS provides a scalable and adaptive framework for real-time detection, analysis, and mitigation.

The proposed architecture advances the state of defensive security in several ways. First, it offers a structured approach to integrating LLMs into cybersecurity workflows through strict schema validation, prompt sanitisation, and policy-governed autonomy. Second, the dual-layer organisation enables DMAS to correlate telemetry across disparate hosts, providing enhanced visibility and contextual reasoning beyond traditional EDR or SIEM systems. Third, the design incorporates strong privacy and governance safeguards, including data minimisation, human oversight, and separation of duties, in alignment with established frameworks such as the NIST AI RMF and the NIST Privacy Framework.

The proof-of-concept (PoC) implementation demonstrates the feasibility of DMAS in a safe, benign environment, showing its ability to perform structured anomaly analysis, generate explainable LLM-derived insights, enforce policy-compliant actions, and adapt to evolving telemetry patterns. Evaluation results highlight promising detection effectiveness, low-latency operation, strong resilience to adversarial perturbations, and high interpretability for human operators.

Several limitations remain, including partial dependence on LLM reasoning, the inability to fully eliminate adversarial evasion at the edge, and the need for long-term field trials to evaluate system robustness in real-world deployments. Future work should explore federated learning approaches for distributed adaptation, integration of model attestations for LLM safety guarantees, and the use of multi-LLM committees for redundancy and consistency checking.

Overall, DMAS represents a balanced and forward-looking defensive architecture capable of countering the expanding landscape of AI-assisted cyber threats while maintaining privacy, accountability, and operational transparency. Its modular design and governance-aware principles make it a promising foundation for future autonomous cyber defence research.

Acknowledgment

The author extends sincere gratitude to the individuals and communities who contributed indirectly to the development of this work. Special thanks to OpenAI's ChatGPT, which served as a discussion partner throughout the conceptualisation, refinement, and structuring of the Defensive Multi-Agent AI System (DMAS) framework. The iterative dialogue and critical questioning significantly strengthened the clarity, completeness, and scientific presentation of the ideas in this paper.

The author also acknowledges the open communities of AI safety, security engineering, and distributed systems research whose publicly available resources helped shape the methodologies used in this project.

A DMAS Architecture Diagram

Figure 1 illustrates the layered structure of the Defensive Multi-Agent AI System (DMAS). The diagram shows relationships between Edge-Agents, the Core-Agent Swarm, and the control/governance layers.

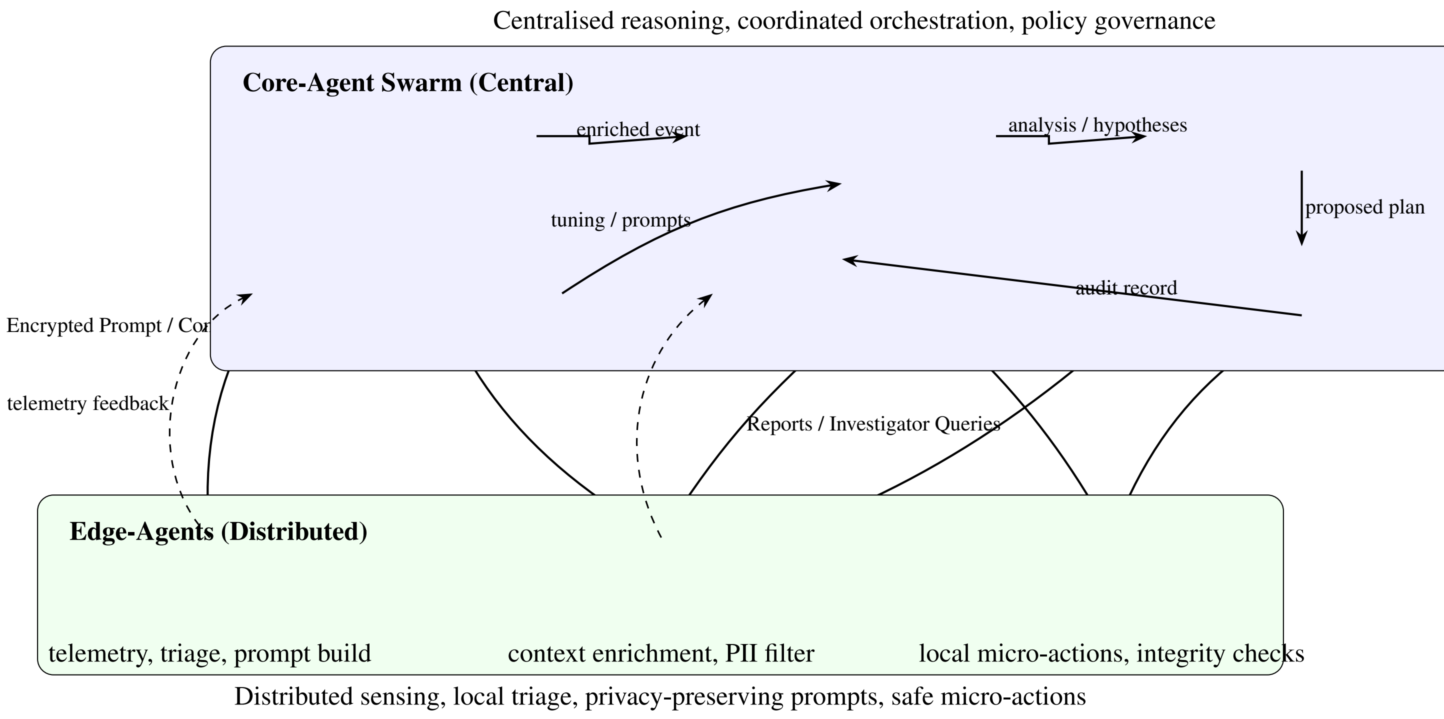
Centralised reasoning, coordinated orchestration, policy governance
Core-Agent Swarm (Central)
enriched event
analysis / hypotheses
proposed plan
tuning / prompts
audit record
Encrypted Prompt / Cor
telemetry feedback
Reports / Investigator Queries
Edge-Agents (Distributed)
telemetry, triage, prompt build
context enrichment, PII filter
local micro-actions, integrity checks
Distributed sensing, local triage, privacy-preserving prompts, safe micro-actions

Figure 1: High-level architecture of DMAS showing data flow, agent interactions, and hierarchical control.

B Edge-Agent Pseudocode

```
function edge_agent_loop():
    event = collect_telemetry()
    triage = local_triage(event)
    context = enrich(event)
    prompt = build_prompt(event, triage, context)
    response = send_to_core(prompt)
    action = response["approved_action"]
    execute(action)
    report_feedback(action)
```

C Core-Agent Reasoning Pipeline

```
function core_pipeline(prompt):
    event = aggregator.normalize(prompt)
    enriched = aggregator.enrich(event)

    analysis = llm_analyst.reason(enriched)
    plan = orchestrator.build_plan(analysis)
    approved = policy_engine.evaluate(plan)

    return approved
```

D Example Structured Prompt

```
{
  "event_type": "process_anomaly",
  "triage_score": 0.82,
  "context": {
    "host": "container-xyz123",
    "process_tree": ["init", "python", "unknown_proc"]
  },
  "constraints": {
    "output_format": "JSON",
    "max_tokens": 350,
    "forbidden_ops": ["delete","shell"]
  }
}
```

E Example Core-Agent Output

```
{
  "hypotheses":[
    {"id":"h1","desc":"Suspicious process", "confidence":0.71}
  ],
  "recommended_actions":[
    {"id":"a1","type":"throttle","target":"container"}
  ],
  "additional_queries":[
    "collect_recent_process_logs"
  ]
}
```

Glossary

Edge-Agent — Lightweight defensive agent deployed at endpoints or network edges responsible for sensing, triage, and limited autonomous actions.

Core-Agent Swarm — Centralised cluster of specialised AI agents that perform reasoning, correlation, planning, policy enforcement, and reporting.

LLM Analyst Agent — Structured reasoning module using LLMs under strict schema and safety constraints.

Policy Decision Agent — Governance gate that ensures actions comply with organisational and regulatory requirements.

DMAS — Defensive Multi-Agent AI System, a two-layer cyber-defence architecture introduced in this paper.

Micro-action — Low-risk, reversible defensive action executed locally at the edge.

PoC — Proof-of-Concept implementation used for validation.

Author Contributions

The primary author conceptualised the Defensive Multi-Agent AI System (DMAS), developed the system architecture, designed the threat model, implemented the proof-of-concept code, and prepared the manuscript.

ChatGPT (OpenAI) contributed through iterative discussion, refinement of concepts, structural organisation, technical drafting of LaTeX components, diagram descriptions, literature mapping, and verification of publicly available references. All final decisions, designs, and interpretations are those of the primary author.

References

[1] Google Cloud Threat Intelligence, "Threat actor use of ai: Latest findings," 2025, accessed: 2025-11-22. [Online]. Available: https://cloud.google.com/blog/products/identity-security/threat-actor-use-of-ai-latest-findings

[2] Microsoft Threat Intelligence, "Generative ai and security: Understanding the risks," 2024, accessed: 2025-11-22. [Online]. Available: https://www.microsoft.com/en-us/security/blog/2024/02/15/generative-ai-and-security-understanding-the-risks/

[3] A. Delamotte, V. Kamluk, and G. Shapiro, "Prompts as code: Llm-enabled malware analysis," 2025, accessed: 2025-11-22. [Online]. Available: https://www.sentinelone.com/labs/prompts-as-code-embedded-keys-the-hunt-for-llm-enabled-malware/

[4] MITRE Corporation, "Mitre att&ck framework," 2015, accessed: 2025-11-22. [Online]. Available: https://attack.mitre.org/

[5] N. I. of Standards and Technology, "Nist cybersecurity framework (csf) version 1.1," 2018, accessed: 2025-11-22. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/CSWP/NIST.CSWP.04162018.pdf

[6] Microsoft Corporation, "Microsoft defender for endpoint documentation," 2025, accessed: 2025-11-22. [Online]. Available: https://learn.microsoft.com/en-us/defender-endpoint/

[7] E. N.V., "Elastic security: Open xdr platform," 2025, accessed: 2025-11-22. [Online]. Available: https://www.elastic.co/security

[8] Kaspersky Global Research and Analysis Team, "Machine learning and artificial intelligence in cybersecurity," 2024, accessed: 2025-11-22. [Online]. Available: https://www.kaspersky.com/resource-center/threats/machine-learning-ai-in-cybersecurity

[9] Google Cloud Security, "Ai-driven threat detection on google cloud," 2024, accessed: 2025-11-22. [Online]. Available: https://cloud.google.com/security/ai

[10] Microsoft Security, "Automated investigation and response (air) in microsoft defender," 2019, accessed: 2025-11-22. [Online]. Available: https://learn.microsoft.com/en-us/defender-endpoint/automated-investigations

[11] Palo Alto Networks, "Cortex xsoar documentation," 2025, accessed: 2025-11-22. [Online]. Available: https://docs.paloaltonetworks.com/cortex/xsoar

[12] Splunk Inc., "Splunk soar documentation," 2025, accessed: 2025-11-22. [Online]. Available: https://docs.splunk.com/Documentation/SOAR

[13] D. Camacho, E. Chiner, and J. Gonzalez-Calderon, "Multi-agent systems for cybersecurity: A review," *IEEE Access*, vol. 6, pp. 3293–3308, 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8247233

[14] N. I. of Standards and Technology, "Nist privacy framework: A tool for improving privacy through enterprise risk management," 2020, accessed: 2025-11-22. [Online]. Available: https://www.nist.gov/privacy-framework

[15] ——, "Zero trust architecture (sp 800-207)," 2020, accessed: 2025-11-22. [Online]. Available: https://csrc.nist.gov/publications/detail/sp/800-207/final

[16] Google DeepMind and Google Cloud, "Securing ai systems: Best practices for mitigating abuse," 2024, accessed: 2025-11-22. [Online]. Available: https://cloud.google.com/security/products/ai-security

[17] Microsoft Corporation, "Microsoft sentinel documentation," 2025, accessed: 2025-11-22. [Online]. Available: https://learn.microsoft.com/en-us/azure/sentinel/

[18] N. I. of Standards and Technology, "National cybersecurity testing and training," 2024, accessed: 2025-11-22. [Online]. Available: https://www.nist.gov/programs-projects/national-cybersecurity-testing-and-training

[19] ——, "Ai risk management framework (ai rmf 1.0)," 2023, accessed: 2025-11-22. [Online]. Available: https://www.nist.gov/itl/ai-risk-management-framework

[20] OECD, "Oecd principles on artificial intelligence," 2019, accessed: 2025-11-22. [Online]. Available: https://oecd.ai/en/ai-principles