INTRUSION RESPONSE THROUGH DEEP PACKET INSPECTION USING MULTI-AGENT SYSTEMS

By

James Williams

May, 2026

Director of Thesis: Sohan Gyawali, Ph.D.

Major Department: Technology Systems

ABSTRACT

As cyber threats continue to increase in both sophistication and frequency, alongside the growing reliance on cloud-based data storage and networked systems, the need for robust and adaptive security solutions has become increasingly critical. Recent advancements in artificial intelligence (AI) have demonstrated significant potential in automating complex tasks and enabling dynamic, context-aware decision-making, making AI a natural fit for modern cybersecurity applications. By integrating AI-driven techniques into cybersecurity systems, organizations can proactively identify and intercept potential attacks, such as network-based intrusions, before they compromise devices, services, or access points. This thesis proposes a multi-agent intrusion response architecture that integrates LLM-assisted reasoning with traditional classification models to detect and respond to malicious network traffic. The system operates inline on the endpoint's gateway, inspecting traffic as it passes through. To evaluate feasibility and effectiveness, the architecture is tested in a controlled network testbed using

multiple Raspberry Pi devices configured as attacker and database nodes and a laptop endpoint, with the multi-agent design compared against a single-agent baseline. Performance was assessed using response latency, detection outcomes, isolation actions, and the quality of generated incident logs. The results aim to characterize whether multi-agent orchestration improves intrusion response reliability and decision quality under near real-time constraints while remaining practical for deployment on dedicated hardware.

INTRUSION RESPONSE THROUGH DEEP PACKET INSPECTION USING MULTI-AGENT SYSTEMS

A Thesis

Presented to the Faculty of the Department of Technology Systems

East Carolina University

In partial Fulfillment of the Requirements for the Degree

Master of Science in Information and Cybersecurity Technology

By

James Williams

May, 2026

Director of Thesis: Sohan Gyawali, Ph.D.

Co-Director of Thesis: Te-Shun Chou, Ph.D.

Thesis Committee Members:

Biwu Yang, Ph.D.

Table of Contents

Page

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1. INTRODUCTION

In today's increasingly interconnected digital landscape, cyberattacks are more frequent, sophisticated, and damaging. A wide variety of attack vectors, ranging from phishing emails and malicious links to advanced malware and network-based exploits, pose significant risks to individuals, organizations, and critical infrastructure. Even seemingly minor user actions, such as clicking on an unsafe link or downloading content from an unverified source, can result in severe security incidents. These incidents often lead to the exposure of sensitive personal and organizational information, including financial data, identity credentials, and proprietary resources, which may then be exploited for fraudulent, disruptive, or destructive purposes. Traditional cybersecurity systems struggle to keep pace with this evolving threat landscape. Signature-based and rule-driven intrusion detection mechanisms are inherently reactive and depend on prior knowledge of attack patterns (Adnan et al., 2025). As attackers increasingly employ encryption, obfuscation, and polymorphic techniques to evade static defenses, these conventional approaches become insufficient for timely and accurate threat detection. This limitation is particularly evident in modern network environments, where encrypted traffic obscures payload visibility, and novel attack behaviors frequently emerge without recognizable signatures. Effective intrusion response is fundamentally dependent on timely, accurate, and context-aware decision-making. To address the shortcomings of traditional systems, there is a growing need for intelligent and adaptive defense mechanisms that can reason about ambiguous signals, correlate multiple sources of evidence, and automate response actions at scale. Recent advancements in artificial intelligence, and specifically Large Language Models (LLMs), present a promising avenue for enhancing intrusion detection and response capabilities. LLMs offer the ability to perform contextual reasoning, protocol-aware interpretation, and threat intelligence

correlation, enabling security systems to move beyond isolated alerts toward coherent and explainable response decisions. The architecture explored in this thesis integrates multiple complementary components, including multiclassification models, LLM-assisted reasoning, and continuously updated threat intelligence. Multiclassification models provide a scalable mechanism for distinguishing between benign and malicious network behaviors across diverse traffic patterns. However, machine learning models alone may produce ambiguous or conflicting predictions, particularly when encountering novel or partially observed threats. An LLM-based multi-agent architecture addresses this limitation by incorporating contextual reasoning and external intelligence, reducing false positives and false negatives while improving the reliability of intrusion response workflows. Finally, effective intrusion response required continuous adaptation to emerging threats. By leveraging retrieval-augmented generation, otherwise known as RAG, and multi-agent LLM architectures that incorporate external datasets and knowledge sources such as MITRE ATT&CK, malware tracking platforms, and open threat intelligence feeds, the proposed system maintains awareness of evolving attacker techniques, malware families, and vulnerabilities. This continuous intelligence ingestion ensures that detection and response capabilities remain current and effective over time. Collectively, these capabilities provide the necessary foundation for modern intrusion response by enabling visibility into, accurate classification and prioritization of heterogeneous threats, contextual understanding through reasoning and intelligence correlation, verification through dynamic analysis, and adaptability through continuous knowledge integration. By transforming raw detection signals into actionable insights, the proposed approach supported faster, more confident, and more effective defensive actions, which are essential in contemporary cybersecurity operations.

1.1 Statement of the Problem

Given the recent popularity of multi-agent systems in cybersecurity, there is growing interest in their potential to improve intrusion detection and response through distributed reasoning and agent collaboration. Traditional security systems often rely on static rules or single-model approaches that struggle to adapt to evolving threats and complex attack behaviors. While multi-agent architectures, particularly those incorporating large language models, promise improved contextual awareness and adaptability (Adnan et al., 2025). Existing research shows mixed results regarding their effectiveness and reliability in real-world environments. Furthermore, challenges related to coordination, computational cost, and evaluation make it unclear when these systems provide a meaningful advantage. As a result, further investigation is needed to assess the practical value and limitations of intrusion-focused multi-agent systems in modern cybersecurity contexts against single-agent systems.

1.2 Research Objectives

The primary objective of this research was to evaluate the feasibility, effectiveness, and real-world deployability of a multi-agent LLM-assisted intrusion response system coupled with deep packet inspection. Specifically, the study assessed whether LLM-based agents could collaboratively identify, analyze, and respond to security incidents under near real-time constraints, and whether the proposed multi-agent architecture improves intrusion response performance compared to a single-agent baseline. Finally, the research evaluated deployment practicality by examining computational requirements, scalability across network environments, and overall cost-effectiveness to determine whether the approach was sustainable for continuous enterprise intrusion response.

1.3 Research Questions

RQ1: How feasible was the proposed multi-agent architecture for intrusion response?

HO1 (Research Hypothesis): Large language models integrated into a multi-agent intrusion response framework would demonstrate sufficient contextual reasoning, response speed, and operational stability to be considered feasible for real-time intrusion response.

HO1 (Null Hypothesis): Large language models integrated into a multi-agent intrusion response framework would not demonstrate sufficient contextual reasoning, response speed, or operational stability to be considered feasible for real-time intrusion response.

RQ2: How effective is the proposed multi-agent LLM for intrusion response?

HO2 (Research Hypothesis): A multi-agent intrusion response system powered by large language models would achieve higher detection accuracy, faster response times, and improved adaptability to diverse cyber threats compared to single-agent baseline intrusion response mechanisms.

HO2 (Null Hypothesis): A multi-agent intrusion response system powered by large language models would not achieve significantly higher detection accuracy, faster response times, or improved adaptability compared to single-agent baseline traditional intrusion response mechanisms.

1.4 Significance of the Study

The results of this study aimed to present a multi-agent intrusion detection and response system capable of identifying and mitigating threats prior to their execution on endpoint devices.

The proposed system was designed with the potential to provide additional support for intrusion detection and response capabilities at the network perimeter. Furthermore, the findings of this study were expected to contribute value to both academic research and practical cybersecurity applications. From an academic perspective, this work extended the existing body of literature on AI-assisted cybersecurity by demonstrating how collaborative intelligent agents can be applied not only to threat detection but also to real-time intrusion response and containment. From an applied perspective, the proposed framework offers a scalable and practical approach that can enhance enterprise security operations, protect sensitive data, and reduce the impact of sophisticated cyberattacks. Ultimately, this research supported the development of more intelligent, proactive, and resilient security systems capable of safeguarding modern digital ecosystems in an era characterized by increasingly AI-driven cyber threats.

1.5 Statement of Purpose

The purpose of the study was to investigate and develop an adaptive, multi-agent intrusion response architecture driven by LLMs to enhance cybersecurity defenses in modern digital environments. This study was motivated by the increasing complexity and frequency of novel cyber threats, particularly those augmented by artificial intelligence tools capable of automating reconnaissance, exploitation, and malware generation. Traditional security systems often rely on static rules or narrowly scoped detection mechanisms, which limit their effectiveness against evolving and intelligent adversaries. By leveraging a coordinated set of intelligent agents capable of real-time analysis, decision-making, and response, this research sought to address these limitations and provide a more resilient security architecture. The overarching goal was to design a deployable, last-resort defensive mechanism that could assist

individuals and organizations in mitigating cyberattacks, preserving data privacy, and maintaining system integrity in scenarios where conventional defenses fail.

1.6 Scope and Limitations

The scope of this research was centered on the use of LLMs within the context of intrusion detection and response, specifically focusing on their ability to detect, analyze, and decompose network-based intrusions. The study explored how LLMs and multi-agent architectures could enhance intrusion response mechanisms through adaptive reasoning and automated decision-making. The analysis was primarily conceptual and comparative in nature, emphasizing system design, feasibility, practicality, and comparison against single-agent systems.

CHAPTER 2. REVIEW OF LITERATURE

Due to the increasing scale and sophistication of modern cyber threats, reliance on a single-agent LLM for detecting novel attacks may be insufficient. At the same time, the design and deployment of multi-agent architectures capable of reasoning about diverse and evolving attack behaviors introduce additional complexity. Although the literature examining both approaches is extensive, this chapter is focused on single-agent and multi-agent LLM architectures for intrusion detection, established methods of intrusion detection, and intrusion response strategies, including approaches for mitigating and containing detected threats.

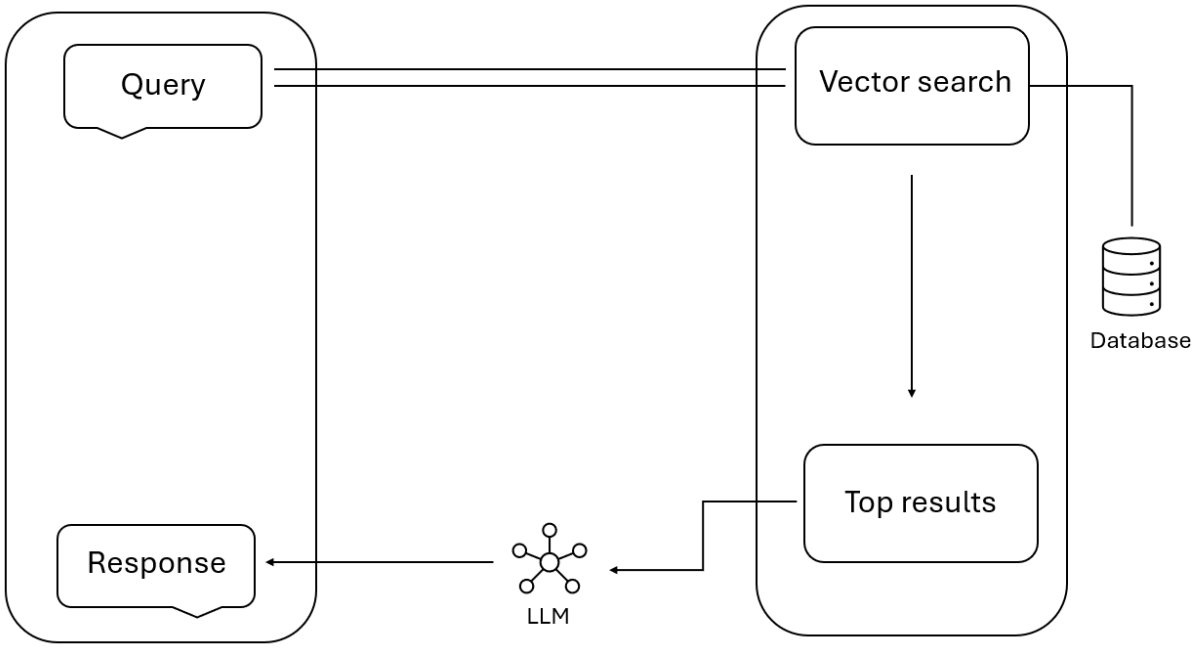
2.1 Retrieval-Augmented and Multi-Agent LLM Architectures

Large language models are increasingly explored as tools for enhancing cybersecurity operations, particularly in threat detection, incident response, and automated analysis of security data. Retrieval-Augmented Generation (RAG) has emerged as a foundational technique for improving LLM outputs by integrating external retrieval mechanisms that supply up-to-date, domain-specific information at inference time. Aquino (2025) describes RAG as a practical approach for grounding generation in retrieved evidence, enabling models to incorporate current and specialized information without continuous retraining. This grounding is especially important in cybersecurity, where inaccurate outputs could lead to incorrect defensive actions or wasted analyst effort. Because cybersecurity response is a high-stakes domain, recent work argues that RAG-based generative systems should be evaluated using structured certification-oriented criteria (e.g., risk-aware assessment of reliability and failure modes) rather than treated as inherently trustworthy simply because retrieval is present (Shan, 2024). Manor et al. (2024) propose a RAG-like mechanism for network packets, using autoencoders to generate a latent-space representation that supported retrieving similar flows and returning human-readable

context for downstream LLM analysis, demonstrating feasibility for attack classification when raw packet structure does not embed well under standard LLM assumptions.

Figure 1

Simple RAG Model



Although RAG improves factual grounding relative to standalone LLMs, its performance still depends on the complexity and modality of the retrieval task. Multimodal RAG variants, for example, have been shown to outperform traditional RAG in scenarios that require both visual and textual understanding (Mei et al., 2025). Research on multi-agent RAG similarly suggests that coordinated agent architectures could outperform strong single-agent baselines. Nguyen et al. (2025) show that MA-RAG, a collaborative multi-agent RAG framework, outperforms standalone LLMs and competitive RAG baselines across multiple benchmarks, indicating that agent collaboration could improve retrieval planning and reasoning. Beyond retrieval, results from multi-agent systems in other reasoning-intensive tasks suggest similar benefits: Xu et al.

(2024) report that their Agent4Debate framework achieves human-level performance in competitive debate settings, surpassing baseline single-agent LLM performance in both automated and expert evaluations.

2.2 Reasoning and Post-Detection Agents in Intrusion Detection

LLM-assisted intrusion detection has emerged as a promising approach for improving the identification and interpretation of malicious network activity through automated reasoning and contextual analysis. In operational settings, LLMs could augment traditional detection pipelines by summarizing evidence, contextualizing alerts, and assisting analysts with interpretation. However, reliability, scalability, and model security concerns remain important considerations when deploying LLMs in production environments. Several studies report strong results from hybrid approaches that combine classical machine learning classifiers with LLM-assisted analysis. Recent IDS designs increasingly pair classical classifiers with retrieval-based LLM interfaces; for example, a chatbot-assisted IDS couples machine-learning traffic detection with a RAG component to retrieve supporting context for user queries and produce analyst-facing explanations of network events (LI et al., 2025). RAG-driven security monitoring also extends beyond packet inspection; Dark Watchdog demonstrates a retrieval-guided pipeline for near-real-time detection and contextual analysis of leaked data in dark web forums, illustrating how retrieval could enrich security conclusions with relevant supporting evidence (Hung et al., 2024). Sudheer et al. (2025) report in Demo: A Real-time Multi-Agent Network Attack Detection and Incident Response System that a detection score of 0.993 was achieved using classical machine learning classifiers combined with LLM-assisted analysis for network attack detection and incident response. Complementary work indicates that LLMs adapted specifically for network traffic analysis can further improve performance when the model is trained or tuned on traffic

representations; for example, TrafficLLM reports F1-scores approaching 0.99 across multiple traffic-analysis scenarios (Cui et al., 2025). At the same time, multiple studies suggest that general-purpose LLMs may be less effective as primary detectors but more effective as post-detection reasoning components. Ikbarieh et al. (2025) report that LLMs are ineffective as primary intrusion detection reasoners in IoT and IIoT networks, but can provide value as post-detection reasoning and mitigation agents. In IoT settings, lightweight fine-tuning approaches (e.g., parameter-efficient methods) have been paired with RAG to improve generalization, retrieving semantically similar examples to enrich prompts and enabling partial zero-shot classification of previously unseen attack types under tight token and resource constraints. (Sudasinghe et al., 2025). Zhang et al. (2024) similarly show that pre-trained LLMs can perform intrusion detection without fine-tuning when augmented with in-context learning strategies. In addition, research on explainable intrusion detection indicates that LLMs may contribute most when used to generate explanations and interpretability layers, rather than as the sole decision mechanism (Houssel et al., 2024). Evidence on whether multi-agent architectures improve intrusion detection performance is mixed. A systematic analysis of multi-agent intrusion detection reports that multi-agent designs do not consistently improve detection accuracy across studies, despite potential benefits for coordination, coverage, and resilience (Saeed et al., 2020). Taken together, these findings suggest that multi-agent designs may be most compelling when they improve operational properties, such as interpretability, workflow integration, and response coordination, rather than when they are assumed to yield uniform gains in raw detection accuracy.

2.3 LLM-Based Post-Detection Incident Response

LLM-assisted intrusion response leverages LLMs to support how security systems interpret alerts and recommend response actions. By combining detection signals with contextual reasoning and external knowledge, LLMs can help explain attack behavior, prioritize incidents, and suggest mitigation steps, potentially reducing analyst workload and improving response consistency. Several studies explore LLMs as response components rather than primary detectors. Hu et al. (2024) present MySQL-Pot: A LLM-Based Honeypot for MySQL Threat Protection, which evaluates an LLM-assisted honeypot that intercepts SQL injection attempts and routes payloads to an LLM for analysis and response generation. Beyond database-focused honeypots, TrapLLM presents a log-based LLM honeypot framework designed for heterogeneous attack logs and deployed in a production network, reporting large-scale capture and automated categorization of real attack behaviors while emphasizing operational requirements such as timely parsing and scalable response generation (Zeng et al., 2025). Lin et al. (2025) present IRCopilot and report that standalone LLMs perform poorly on incident response tasks due to issues such as hallucinations, privacy constraints, and context loss. Their results indicate that structured multi-agent orchestration improves task completion and the practical utility of LLM-supported response workflows. Kramer et al. (2025) evaluate whether LLMs can support incident summarization, a late-stage activity in incident response. Using 50 real-world incidents and evaluations from 18 professional security analysts, they find that autonomous LLM summaries are preferred less often than human-authored summaries, primarily due to omitted details and factual inaccuracies. In contrast, when LLMs are used as drafting assistants, analysts more frequently prefer AI-assisted summaries and report improvements in readability and consistency, reinforcing the value of LLMs as collaborative tools rather than fully

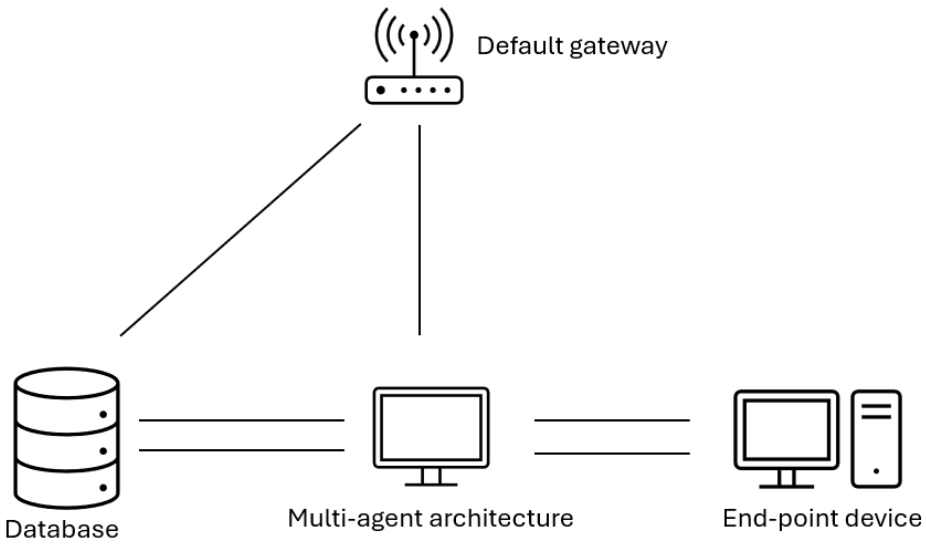
autonomous responders. Building on this perspective, Ismail et al. (2025) evaluate an agentic, LLM-driven security orchestration, automation, and response (SOAR) platform designed to automate post-detection incident response in Security Operations Centers. In a pilot deployment integrated with a Wazuh SIEM, the system autonomously executes multi-step mitigation workflows, including enrichment, mapping to the MITRE ATT&CK framework, response generation, and tool-level execution. The authors report reduced response complexity by compressing a 38-step playbook into 10 higher-level LLM-generated actions, while maintaining appropriate sequencing and alignment with established frameworks. At the same time, they note that metrics such as mean time to respond and end-to-end mitigation success rates remain future work, emphasizing validated feasibility rather than proven superiority over existing SOAR platforms. Post-detection enrichment can also be applied to rule-based IDS output, Lee et al. (2025) translate Snort rules into human-readable summaries and employ a RAG framework grounded in the MITRE ATT&CK knowledge base to map these descriptions to corresponding tactics and techniques. This approach aims to reduce hallucinated mappings while improving analyst interpretability and the overall usefulness of response recommendations.

CHAPTER 3. METHODOLOGY

This section outlines the research design, framework, and analytical approach used to evaluate and compare AI-driven cybersecurity systems. The aim of this study was to evaluate whether the proposed multi-agent architecture could support real-time detection, response, isolation, and prevention of malware within the network and on the host machine. The methodology focused on examining how the multi-agent architecture responds to threats. Much of this thesis was carried out through experimental implementation, identified trends and advantages that certain architectures had over others, and combining different methods of detection and response. A multi-agent RAG pipeline, network intrusion detection, and agentic intrusion response pipeline were created, capable of taking action against detected threats to end-device security. Multiple versions and system configurations were tested, with attacking devices both inside and outside the network. In these scenarios, attackers launched unauthorized attempts to enter the network and to compromise the host machine. Chapter 3, Methodology, includes the research design, data collection, data analysis, detection testing, and response testing procedures. The code repository for the architecture can be found on GitHub following the link https://github.com/Jaywil27/Orchestrated-Multi-Agent-LLM-Framework-for-Intrusion-Detection-and-Response-Automation.

Figure 2

Overarching Network Testbed for the Architecture



3.1 Research Design

Figure 2 showed one of the many possible configurations used to evaluate the performance of the architecture against threats to the endpoint device. The database acted as a data source from which the endpoint device gathers information and is connected to the default gateway in the event that the endpoint device issues a request the database cannot fulfill. The database served as a device for fulfilling requests from the endpoint device. The multi-agent architecture was positioned between the endpoint device and the database and acted as the endpoint device's default gateway. Its purpose is to filter out malicious traffic attempting to access the endpoint device. This overarching setup was used as the network testbed for future experiments involving the multi-

agent architecture under different attack styles. The testbed was designed to evaluate the multi-agent architecture's ability to detect malicious or abnormal packets in near real time and to isolate these threats. The multi-agent architecture was positioned between the database and the endpoint to intercept potentially compromised packets sent by an infected device. A Raspberry Pi, was used as the attacker, another Raspberry Pi served as the database, while a laptop functioned as the endpoint device. The multi-agent architecture runs on a dedicated computer. The general flow through the pipeline was detection layer to response layer. A Raspberry Pi was chosen over a laptop or other device as the attacker to provide a more realistic testbed when scaling to multiple attackers targeting the endpoint device, while also ensuring that device configuration does not become overly complex. A Raspberry Pi was also chosen as the database to better mimic a real-world deployment rather than virtualizing or simulating the device. A dedicated computer was selected to run the multi-agent architecture due to the expected computational demands of the system and its requirement for minimal latency; therefore, a cloud-based deployment was not considered. The dedicated computer used 32 GB of ram and a 12 GB 3060 Nvidia GPU with a Intel I7 12700K CPU. Furthermore, LLM inference was performed locally on the dedicated machine using a locally hosted model that being GPT-OSS:20b to further reduce latency. Multiple configurations of the multi-agent architecture were evaluated to compare detection and response effectiveness. Additionally, the multi-agent architecture was compared to a single-agent architecture to measure effectiveness in key areas such as intrusion detection and intrusion response. In terms of intrusion response, the single-agent architecture responds to threats in a similar manner to the multi-agent architecture by attempting to secure the endpoint and generating a log summarizing the attack.

Overall, several detection and response methods were evaluated, including:

- Creation of attack logs summarizing events
- Isolation of detected threats
- Time taken to respond to detected threats
- Decision-making processes used by the agent(s) when isolating a threat
- Rationale for isolating a given threat
- Quality of summarized attack logs

This research design directly supported the evaluation of the proposed research questions and associated hypotheses. To address RQ1, the feasibility of the multi-agent architecture for intrusion response was assessed through controlled experimental deployment, focusing on contextual reasoning capability, response speed, and operational stability during real-time attack scenarios. Feasibility was evaluated based on the architecture's ability to consistently detect, interpret, and respond to intrusions without degradation in performance under repeated and varied attack conditions. To address RQ2, the effectiveness of the proposed multi-agent LLM-based intrusion response system was evaluated through comparative experimentation against a single-agent architecture under identical network and attack conditions. The single-agent baseline consists of a unified LLM-driven pipeline responsible for both analysis and response generation without task decomposition or agent specialization. Unlike the proposed multi-agent architecture, this approach does not incorporate disagreement handling, role-specific reasoning, or collaborative decision-making. Detection accuracy, response latency, adaptability to different attack styles, and quality of response actions are used as primary evaluation criteria. Performance improvements across these dimensions are interpreted as evidence supporting the research hypotheses, while the absence of such improvements supported the corresponding null

hypotheses. Together, these evaluation criteria provide a comprehensive basis for assessing the effectiveness, responsiveness, and decision-making capabilities of the proposed architecture.

3.2 Environmental Setup

To establish the requirements for testing the proposed architecture, a network testbed was created that incorporated all necessary assets, hardware, and system components. The multi-agent architecture was implemented and deployed on a dedicated computer. To ensure that the endpoint device routed traffic through the dedicated computer, the endpoint was configured to use the dedicated computer as its default gateway, while the dedicated computer was configured to retrieve requested data from the database.

Figure 3

Network Testbed for Attacks

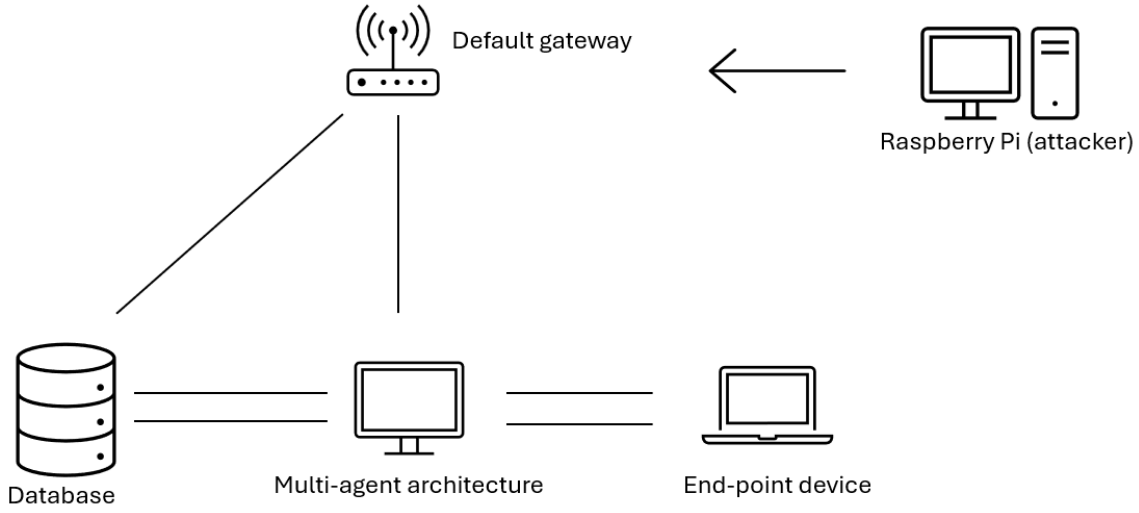


Figure 3 showed the testbed used for executing one of the possible attack scenarios. The attacking device, a Raspberry Pi, was used as the attacker. Another Raspberry Pi served as the database, while a laptop functioned as the endpoint device. The multi-agent architecture runs on a

dedicated computer. In this scenario, the attacker attempted to gain unauthorized access to the network and is successful, after which the attacker can launch attacks against the endpoint device. It is important to clarify that the endpoint device is not on the same network as the multi-agent architecture, as the architecture acted as the endpoint's default gateway in these scenarios. From this point, the attacker is able to infect the database from which the endpoint requests data. At this stage, the endpoint may receive potentially malicious packets, and the attacker can learn of the endpoint's existence through locally stored database logs that verify the endpoint's activity.

Figure 4

Attacker Infects Database to Transmit Malicious Data to Endpoint

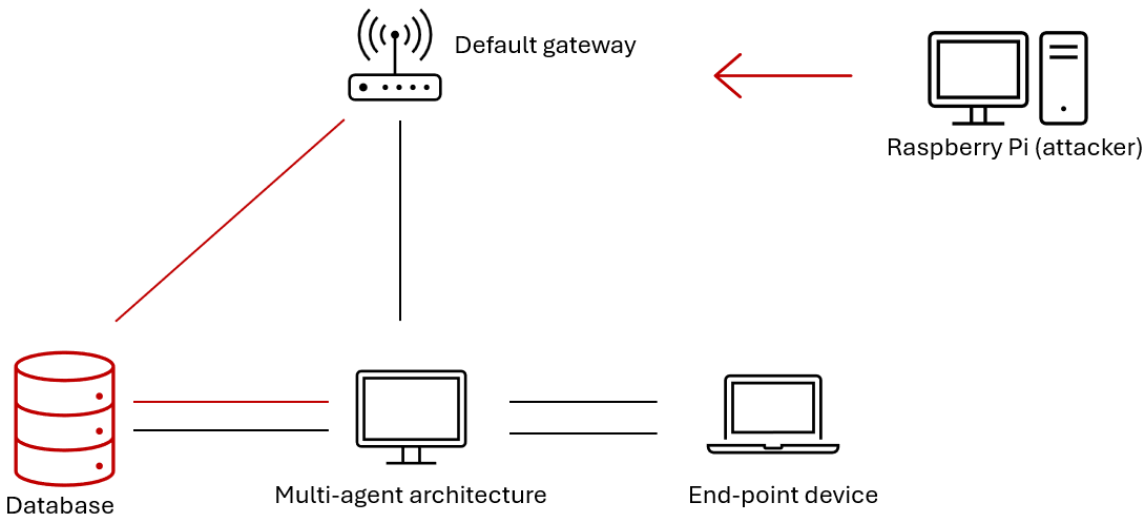


Figure 4 showed the attacker having successfully infected the database from which the endpoint requests data. At this point, any packet sent by the database may be potentially infected. For infected traffic, specially crafted packets were generated using Scapy and transmitted by the

database. These packets contained a flag, that the multi-agent architecture was required to detect as malicious, before the packets reached the endpoint.

3.3 Design of the Architecture

This section outlines the architecture of the multi-agent system. The architecture used multiple agents to perform detection and a GPT-OSS:20B pretrained LLM model for response. Everything starts with the classification agents which will be analyzing the incoming data packets. The architecture used multiple classification agents such as XGBoost, Random Forest Tree, and MLP, trained on analyzing network traffic and classifying that traffic as benign or malicious. Due to the usage of multiple classification agents there may be times where there are two conflicting results with a packet being labeled as benign by one model and malicious by another. When packets are labeled as benign, they are allowed to traverse the network to the endpoint device. If packets are labeled as malicious, the connection is suspended and the endpoint device is alerted that malicious or abnormal packets were detected and can choose whether to allow said connection to resume or drop the connection. The primary objective of the system is to facilitate the tracking and logging of potentially malicious network packets, utilizing AI-based classification techniques for threat detection. While multiple classifiers may operate concurrently, benign traffic is permitted to pass through without restriction. This raises an important consideration for design: how are packets evaluated and classified as benign or malicious? The system supported two user-selectable methods for malicious packet identification. The first method follows conventional intrusion detection practices by examining packet characteristics and determining whether they violate predefined security rules or deviate substantially from expected network behavior. Although LLM responses may be limited to analysis and recommendation, the architecture includes the capability to execute operating

system–level commands through the system terminal or shell to block or terminate connections identified as potentially malicious. These actions may be performed automatically based on system policy or at the discretion of the user. The following will describe the architecture and its systems more in-depth.

3.3.1 Traffic Capture and Feature Extraction

This subsection describes how network traffic is captured and which features are extracted to support the classification agents. Because the machine-learning models require structured inputs, packets are collected into a short sliding buffer or window to enable feature extraction and inference with minimal latency. Network traffic is captured using Wireshark, which generates packet capture (PCAPNG) files for analysis. These capture files are subsequently processed using TShark, the command-line interface of Wireshark, to extract relevant packet-level features and convert the raw traffic data into a structured CSV format suitable for machine learning model training. TShark is executed within a Linux (Ubuntu) environment via the terminal, enabling efficient batch processing of large capture files. The extracted fields are carefully selected to include the necessary attributes required for downstream feature engineering, allowing additional derived features to be computed for model training. A detailed table outlining the extracted features is provided below.

Table 1

Extracted Features

Feature	Why It Is Extracted
frame.time_epoch	Enables temporal analysis such as packet rates, interarrival times, and traffic bursts critical for detecting DoS and scanning behavior.

frame.len	Provided packet size information used to identify abnormal payload patterns and compute traffic volume metrics.
ip.proto	Distinguishes protocol types (TCP, UDP, ICMP), which is essential since different attacks target specific protocols.
ip.src	Allows identification of traffic sources, enabling detection of distributed attacks (e.g., DDoS) or single-source anomalies.
ip.dst	Helps identify targeted systems, useful for detecting focused attacks such as port scans or targeted exploits.
tcp.srcport	Captures source-side communication behavior, useful for identifying spoofing or abnormal client activity.
tcp.dstport	Critical for identifying targeted services and detecting port scanning or service-specific attacks.
udp.srcport	Provided insight into UDP-based communication patterns and helps detect abnormal traffic sources.
udp.dstport	Enables detection of UDP-based attacks targeting specific services or ports.
tcp.flags.syn	Identifies connection initiation attempts, crucial for detecting SYN flood attacks and scanning behavior.
tcp.flags.ack	Indicates normal connection acknowledgment behavior, helping distinguish legitimate traffic from anomalies.
tcp.flags.reset	Captures abrupt connection terminations, often associated with failed connections or malicious probing.

The extracted features form the foundation for the subsequent feature engineering process. A comprehensive set of features is initially derived from the PCAPNG files to facilitate the generation of additional, more informative attributes. This approach enables the construction

of a feature-rich dataset capable of generalizing across a wide range of attack types, domains, and threat vectors, including both internal and external network threats. Building upon the features outlined in the previous table, additional engineered features are introduced to further enhance model performance. These derived features capture higher-level statistical and behavioral characteristics of network traffic, improving the classifiers' ability to distinguish between benign and malicious activity. A detailed table describing these engineered features, along with the rationale for their inclusion, is provided below. All classifiers in the architecture utilize the same feature set. This design choice ensures consistency across models and maintains a balanced, homogeneous input space, enabling fair comparison and effective ensemble integration. Feature selection is guided by their relevance to capturing network behavior and identifying patterns indicative of malicious activity.

Table 2

Additional Features

Feature	Why It Is Extracted
window_start	Defines temporal boundaries for aggregation, enabling consistent time-based analysis of traffic behavior.
packet_count	Measured traffic volume, a primary indicator of flooding attacks such as DoS.
byte_count	Captures total data transfer, helping differentiate between high-volume and low-volume attacks.
avg_pkt_size	Identifies typical packet size patterns, useful for detecting anomalies like unusually small or large packets.
std_pkt_size	Measured variability in packet sizes, which can indicate irregular or crafted traffic patterns.
min_pkt_size	Helps detect unusually small packets often used in scanning or flooding attacks.
max_pkt_size	Identifies unusually large packets that may indicate payload-based attacks.
tcp_count	Quantifies TCP usage, important since many attacks (e.g., SYN floods) target TCP.
udp_count	Tracks UDP traffic volume, useful for detecting UDP floods or amplification attacks.

icmp_count	Identifies ICMP activity, commonly used in ping floods or network reconnaissance.
syn_count	Measured connection initiation attempts, a key indicator of SYN flood and scanning attacks.
ack_count	Reflected normal connection behavior, helping distinguish legitimate traffic from malicious patterns.
rst_count	Indicates abnormal connection resets, often linked to scanning or failed connections.
unique_src_ips	Detects distributed attack patterns by measuring the diversity of traffic sources.
unique_dst_ips	Identifies whether traffic is targeting multiple systems or a single victim.
unique_src_ports	Helps detect randomized or spoofed source behavior.
unique_dst_ports	Critical for identifying port scanning activity across multiple services.
mean_interarrival_time	Captures average packet timing, useful for detecting rapid traffic bursts.
std_interarrival_time	Measured variability in timing, helping identify irregular or automated traffic.
min_interarrival_time	Detects extremely fast packet arrivals typical of high-rate attacks.
max_interarrival_time	Captures delays that may indicate intermittent or stealthy behavior.
packets_per_sec	Measured traffic intensity, a key metric for detecting flooding attacks.
bytes_per_sec	Quantifies data throughput, helping distinguish between volumetric and low-rate attacks.
tcp_ratio	Shows protocol distribution, useful for identifying protocol-specific attacks.
udp_ratio	Helps detect abnormal increases in UDP traffic.
icmp_ratio	Identifies unusual ICMP usage, often linked to reconnaissance or flooding.
syn_ratio	Highlights excessive connection attempts relative to normal traffic, indicating SYN floods.
ack_ratio	Reflected balance in TCP communication, helping identify incomplete or abnormal connections.
rst_ratio	Indicates abnormal termination rates, often associated with scanning or probing.
syn_ack_ratio	Detects imbalance between connection initiation and acknowledgment, a strong indicator of SYN flood attacks.
dst_port_entropy	Measured randomness in targeted ports, critical for identifying port scanning behavior.
attack_label	Provided ground truth for supervised learning, enabling model training and evaluation.

To prevent feature leakage in time-based engineered features, attack patterns are randomized during each model evaluation. This ensures that the models do not learn unintended temporal dependencies, ordering artifacts, or time-based signatures present in the training data, thereby improving generalization to unseen traffic patterns. The additional features are derived from the TShark-extracted data using a Python-based feature engineering pipeline. This process operates on the previously extracted packet-level attributes to generate higher-level, more informative features, such as protocol ratios (e.g., TCP ratio), destination port entropy, and other statistical metrics that capture underlying network behavior. These derived features enhance the model's ability to identify complex patterns associated with malicious activity. A key component of the feature engineering process is the use of a window-based feature representation. In this system, attack detection is performed at the window level rather than on individual packets, making the choice of window size a critical factor in classification performance. The window size defines the number of packets observed within a fixed time interval, and all features used for classification are computed from aggregated packet characteristics within each window. This approach enables the capture of short-term behavioral patterns in network traffic, which are often indicative of attack activity. This design is motivated by both efficiency and practicality. In high-throughput network environments, analyzing individual packets or processing large batches can be computationally expensive and inefficient. By aggregating packets into windows, the system reduces computational overhead while preserving meaningful contextual information. Furthermore, window-based analysis provided richer behavioral insight compared to single-packet inspection, resulting in more robust detection performance. All classifiers are trained using the same set of window-derived features, ensuring consistency across models and maintaining a balanced and homogeneous feature space. This uniform representation allows each

classifier to learn from the same underlying data distribution, improving comparability and enhancing overall ensemble performance. The selection of window size has a significant impact on detection effectiveness and may vary depending on the type of attack being targeted. Smaller windows enable faster, more responsive detection and are better suited for capturing short-lived attack bursts, but may introduce noise due to limited context. Conversely, larger windows provide more stable and informative feature representations but may increase detection latency and potentially obscure transient attack behavior. These trade-offs are summarized in the table below.

Table 3

Window Size Information

Window	Pros	Cons	Timeframe
Smaller windows	• Provided more rows • Makes detection more responsive • Capture short bursts better	• Makes each row noisier • May contain too few packets in low-traffic periods • Can make labels less stable	0.05 0.1 0.5
Larger windows	• More stable • Smooth noise	• Can blur attacks into normal traffic	0.5 1.0 1.5 2.0+

For the initial training dataset, a window size of 0.05 seconds was selected. This choice was made to support the detection of multiple attack types while maintaining a high level of responsiveness and low latency during inference. Many network attacks, particularly burst-based attacks, occur over very short time intervals, making a smaller window size effective for

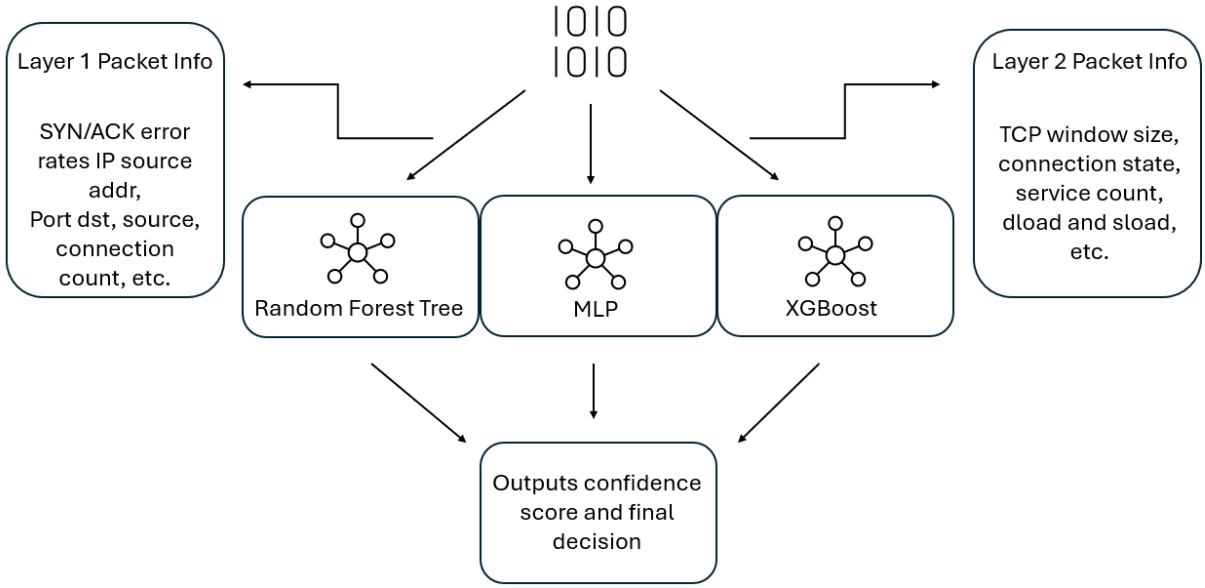
capturing these rapid behavioral changes. It is important to note that the optimal window size may vary depending on the specific attack being targeted. When the system is tailored toward detecting a particular class of attack, the window size should be adjusted accordingly to align with the temporal characteristics of that attack. In such cases, the feature engineering process must also be adapted to ensure that relevant behavioral patterns are effectively captured within the selected window. This alignment between window size and attack dynamics is critical for achieving optimal detection performance.

3.3.2 Multi-Classifier Detection Layer (XGBoost, Random Forest, MLP)

Each classifier in the system is trained using the same dataset and a consistent feature set derived from window-based traffic representations. During inference, all classifiers operate on the same window of packet data, ensuring that each model evaluates identical input regardless of whether another classifier has already predicted the traffic as benign or malicious.

Figure 5

Classifier Detection Layer Overview



This design guarantees that all classifiers contribute equally to the decision-making process and that no potentially relevant feature interactions are overlooked. The classifiers are provided with aggregated window-level features, which encapsulate both packet-level and behavioral characteristics required for accurate inference. Performing inference across all classifiers for every window enables a comprehensive analysis of network traffic, as different models may capture distinct patterns or relationships within the data. The architecture employs three complementary classifiers Random Forest Tree, XGBoost, and a Multi-Layer Perceptron (MLP) to improve detection reliability and reduce both false positives and false negatives. The Random Forest model provided fast, low-latency predictions and performs an initial broad screening of traffic. The XGBoost model offers a more discriminative classification by leveraging complex feature interactions, reducing the likelihood of misclassifying malicious traffic as benign. The MLP classifier introduces an additional independent learning perspective, improving generalization and mitigating bias associated with tree-based models. The use of

multiple classifiers ensures that packet data is analyzed at different levels of abstraction and decision boundaries, resulting in a more robust and accurate detection process. Limiting the architecture to three classifiers balances detection performance with computational efficiency, avoiding excessive overhead while still providing sufficient model diversity. Each classifier produced a numerical output corresponding to a specific traffic class. These outputs are subsequently decoded into their human-readable labels (e.g., benign, DoS, port scan), as defined in the mapping table below. This encoding scheme standardizes classifier outputs and facilitates their integration into downstream components, including the response generation and RAG-based analysis stages.

Table 4

Label Encoding

Encoding	Plain text variant
0	Benign
2	Port scan
1	DoS

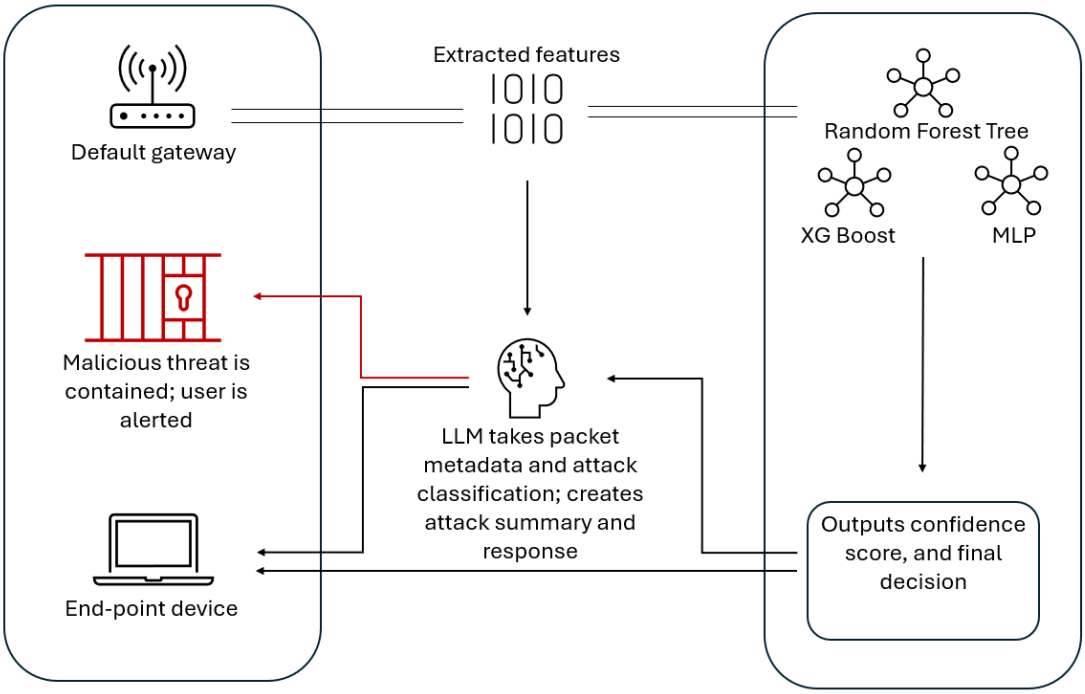
3.3.3 Disagreement Handling and Tie-Breaking Logic

For disagreement handling, the system employs an ensemble voting mechanism. A majority voting strategy was used, whereby if the majority of classifiers identify a window as malicious, the window is classified as an attack and forwarded to the RAG-based system for further analysis along with extracted features. This approach was selected due to its computational efficiency and simplicity, making it well-suited for real-time detection scenarios. Additionally, prioritizing majority agreement helps reduce the likelihood of false negatives,

ensuring that potentially malicious traffic is not overlooked. By erring on the side of caution, the system increases its sensitivity to attack detection while maintaining a manageable computational overhead.

Figure 6

Map of Detection Pipeline



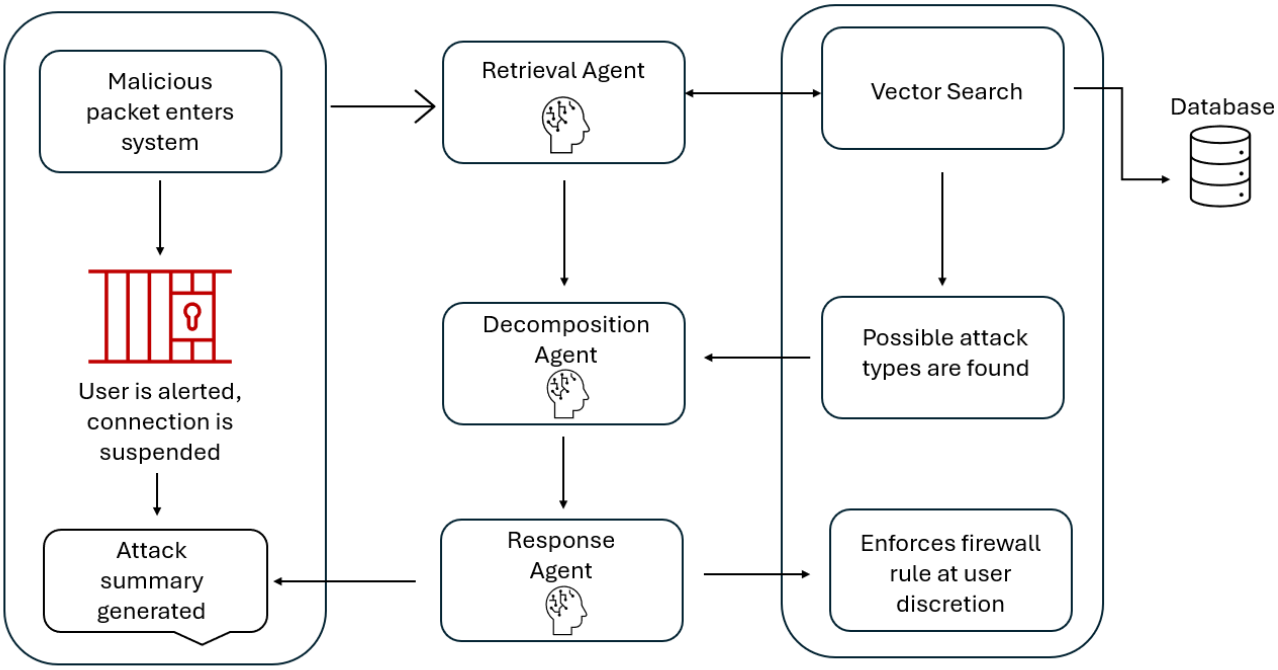
3.3.4 Response and Alerting

When malicious activity is detected by the classifiers, the system forwards the classification results along with relevant metadata derived from the corresponding traffic window to the LLM-based analysis layer. This information is structured as a context package, which includes the extracted feature values, window-level statistics, and the predicted labels produced by each classifier. The Retrieval-Augmented Generation (RAG) system utilizes this context package to query its knowledge base for information related to the identified threat. Based on the retrieved data, the system generates a detailed incident report describing the nature

of the attack, its characteristics, and its potential impact. In parallel, the system searches a dedicated response database containing predefined firewall rules and command-line mitigation strategies to determine appropriate countermeasures.

Figure 7

Multi-agent Response Pipeline



The generated incident report, along with recommended mitigation actions, is then presented to the user. Depending on the system configuration, these actions can either be suggested for manual execution or automatically applied to respond to and mitigate the detected threat.

3.4 Datasets and Knowledge Sources

This research leverages established threat intelligence frameworks and open-source knowledge bases to support intrusion detection and response. Primary among these is MITRE ATT&CK, which provided a structured and widely adopted taxonomy of adversarial tactics,

techniques, and procedures (TTPs). The MITRE ATT&CK framework was used to contextualize detected behaviors, map observed activities to known attack techniques, and support reasoning about potential attacker intent during intrusion response. In addition to the enterprise-focused ATT&CK matrix, this work incorporates domain-specific extensions of the framework, including MITRE ATT&CK for Industrial Control Systems (ICS) and MITRE ATT&CK for IoT. These specialized matrices are used to better represent attack techniques relevant to resource-constrained devices, embedded systems, and cyber-physical environments. Incorporating these variants allows the system to reason about threats that may not be adequately captured by traditional enterprise-centric intrusion detection approaches. To complement the MITRE knowledge base, additional open-source threat intelligence databases and malware tracking repositories are used to provide up-to-date contextual information on emerging threats, known malware families, and active exploitation trends. These sources are queried periodically, either at system startup or at fixed intervals (e.g., every four hours) to ensure that the intrusion response framework maintains awareness of newly disclosed techniques, indicators, and threat behaviors. By combining structured adversary knowledge from MITRE ATT&CK with continuously updated open-source threat intelligence, the proposed architecture supported more informed detection, contextual analysis, and adaptive intrusion response decisions.

CHAPTER 4. EXPERIMENTAL RESULTS

This chapter presents the evaluation methodology for how each classifier was evaluated and the reasons for why these decisions are made. Following the evaluation methodology is the conditions of the testbed and classifier models used, comparative analysis of the classifiers and their detections and the LLM output in comparison to the attacks that were run in the testbed and finally a comparative analysis of the single and multi-agent outputs.

4.1 Evaluation Methodology

The performance of each classifier was evaluated using standard statistical metrics for binary classification. Each traffic class (e.g., DoS, benign, port scan) is treated independently as a binary classification problem, where each sample is assigned a label:

$$y \in \{0,1\}$$

with 1 indicating membership in the target class and 0 indicating non-membership. Predicted labels are compared against ground-truth labels, and performance is quantified using the following metrics.

4.1.1 Confusion Matrix

All evaluation metrics are derived from the confusion matrix:

$$\begin{bmatrix} TN & FP \\ FN & TP \end{bmatrix}$$

where:

- TP(True Positives): correctly classified positive samples
- TN(True Negatives): correctly classified negative samples

- FP(False Positives): negative samples incorrectly classified as positive
- FN(False Negatives): positive samples incorrectly classified as negative

The confusion matrix provided a complete characterization of classifier outcomes and served as the basis for all subsequent performance metrics.

4.1.2 Accuracy

Accuracy measured the overall proportion of correctly classified samples:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Accuracy provided a global measure of model performance and is appropriate when class distributions are balanced and all types of classification errors are considered equally important. However, in intrusion detection contexts, accuracy alone may be insufficient due to potential class imbalance and differing costs associated with false positives and false negatives.

4.1.3 Precision

Precision measured the proportion of predicted positive samples that are truly positive:

$$\text{Precision} = \frac{TP}{TP + FP}$$

Precision reflected the reliability of positive predictions. In the context of network intrusion detection, high precision indicates a low rate of false alarms, which is important for minimizing unnecessary alerts and reducing operational overhead.

4.1.4 Recall (True Positive Rate)

Recall, also referred to as the True Positive Rate (TPR), measured the proportion of actual positive samples that are correctly identified:

$$\text{Recall} = \frac{TP}{TP + FN}$$

Recall is particularly critical in security applications, as it quantifies the ability of the model to detect malicious activity. A low recall indicates that attacks are being missed, which can have significant consequences in real-world deployments.

4.1.5 F1-Score

The F1-score is the harmonic mean of precision and recall:

$$\text{F1-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

The harmonic mean penalizes large disparities between precision and recall, ensuring that both metrics must be high to achieve a strong F1-score. This makes it a suitable metric when both false positives and false negatives are important considerations.

4.1.6 Receiver Operating Characteristic (ROC) and Area Under the Curve (AUC)

The Receiver Operating Characteristic (ROC) curve represents the trade-off between the True Positive Rate (TPR) and the False Positive Rate (FPR) across varying classification thresholds:

$$\text{TPR} = \frac{TP}{TP + FN}, \text{FPR} = \frac{FP}{FP + TN}$$

The Area Under the ROC Curve (AUC) is defined as:

$$\text{AUC} = \int_0^1 \text{TPR}(FPR)\, d(FPR)$$

ROC AUC provided a threshold-independent measure of classifier performance, capturing the model's ability to rank positive samples higher than negative samples. An AUC value of 1.0 indicates perfect separability, whereas a value of 0.5 corresponds to random classification. This metric is particularly useful for evaluating the quality of model confidence scores and comparing classifiers beyond fixed decision thresholds.

4.1.7 Total Classification Error

The total number of misclassified samples is given by:

$$\text{Error Count} = FP + FN$$

This metric provided an absolute measure of classification errors and complements proportion-based metrics such as accuracy. It is particularly useful for understanding the practical impact of model performance in terms of incorrectly classified instances.

4.1.8 Binary Evaluation of Multi-Class Data

Although the dataset contains multiple traffic classes, evaluation is conducted using a one-vs-rest (OvR) approach, where each class is treated as a separate binary classification problem:

$$\text{Class}_i \text{ vs Not Class}_i$$

This approach enables detailed, class-specific performance analysis and is well-suited for intrusion detection tasks, where different attack types may exhibit distinct characteristics and varying levels of detection difficulty.

4.1.9 Rationale for Metric Selection

A combination of complementary metrics is employed to provide a comprehensive evaluation of classifier performance:

- **Accuracy** quantifies overall correctness
- **Precision** measured the reliability of positive predictions
- **Recall** evaluates the ability to detect malicious activity
- **F1-score** balances precision and recall
- **ROC AUC** assesses the quality of probabilistic predictions

Together, these metrics provide a robust and multidimensional assessment of classifier effectiveness in identifying malicious network traffic.

4.1.10 LLM Response Evaluation

This section describes how the multi-agent and single-agent responses were evaluated. To assess the quality of the generated incident summaries and command line interface (CLI) commands chosen as the action to block or mitigate the detected threat, an LLM-as-a-judge approach was used. In this method, multiple LLMs are selected to independently evaluate the quality of each response. The evaluator models used in this study were Claude Sonnet 4.6, ChatGPT 5.4, and Gemini 3. Each model rated the generated responses from the multi-agent and single architectures used in the study on a scale from 0 to 3, where 0 represented poor

performance, 2 represented moderate performance, and 3 represented high performance. Using multiple evaluator models helped reduce reliance on a single model's judgment and provided a broader assessment of response quality. The responses were evaluated using several criteria: accuracy, completeness, relevance, actionability, clarity, and hallucination. Accuracy was included to determine whether the response correctly identified the attack type and accurately reflected the technical details provided in the context package. This metric is important because an incident response that misidentifies the threat or misrepresents the underlying data may lead to incorrect defensive actions. Completeness was used to measure whether the response included all essential components of an incident report, such as a summary of the event, relevant indicators, and mitigation guidance. This criterion was necessary because even a factually correct response may be inadequate if key details are omitted. Relevance was used to assess whether the generated response directly addressed the detected threat and remained grounded in the provided context. This metric was important because security responses must remain focused on the specific event being analyzed rather than introducing unrelated or unnecessary information. Actionability was included to evaluate whether the recommended mitigation steps were practical, technically valid, and capable of being implemented in a real environment. This was a necessary criterion because the value of an incident response system depends not only on its ability to describe a threat, but also on its ability to recommend useful defensive actions. Clarity was used to examine how well the response was structured and how easily it could be understood by a human analyst. This metric was included because incident summaries must be readable and interpretable in order to support timely decision-making. Finally, hallucination was evaluated to determine whether the response introduced unsupported, fabricated, or unverified information not present in the provided context. This criterion was particularly important because

hallucinated details in a cybersecurity setting may mislead analysts and reduce trust in the system's recommendations. Together, these evaluation criteria were selected to provide a balanced assessment of both the technical correctness and practical usefulness of the generated responses. As a result, this evaluation framework made it possible to compare the performance of the single-agent and multi-agent architectures not only in terms of response quality, but also in terms of their reliability and operational value for incident response.

4.2 Detection Results

This section will detail the results of the classifiers on detection. Detection performance was evaluated by training and testing the models on CSV datasets generated from packet capture (PCAP) data representing multiple attack scenarios. A comprehensive training dataset was constructed from a packet capture containing approximately 5 million packets, organized into roughly 60,000 windows of size 0.05 seconds. Of these packets, approximately 3.5 million corresponded to DoS attack traffic and 500 thousand to port scan attacks, while the remaining 1 million represented benign traffic. Although this distribution is imbalanced, it was intentionally designed to emphasize the detection of a broad range of attack behaviors. In practice, this distribution yielded strong performance across the evaluated models. The datasets used for testing were two main sets each containing a different attack, a DoS attack set containing ~1 million packets, organized into ~1000 windows of size 0.05. A port scan attack dataset containing ~44k packets organized into roughly ~1000 windows of size 0.05.

4.2.1 Evaluation of the Detection Mechanism

This section details the evaluation of the detection model each section details the attacks used against the classification model for these classifications the same model that was trained using the 5 million packet dataset was used across all attacks.

Two types of attacks were tested against this model:

- Port scanning
- SYN flood

These two attacks were chosen as in terms of packet count which is a key metric they are opposite of each other where SYN flood attacks are a large amount of packets all in a small timeframe whereas a port scan is a much smaller and quieter attack that has the potential to lay the foundation of a larger attack, due to the nature of a port scan it is more difficult to detect than a SYN flood attack showcasing the models ability to detect opposite styles of attacks.

4.2.2 DoS Attack Data Statistics

This section presents the statistical characteristics of the dataset derived from the SYN flood (DoS) attack packet capture. The dataset was generated to evaluate classifier performance under conditions where only denial-of-service (DoS) attack traffic and benign traffic are present. Attacks were executed at randomized time intervals to better simulate real-world network behavior, where malicious activity does not occur in a strictly periodic or predictable pattern. The use of randomized attack intervals ensures that the resulting traffic captures include a mixture of high-intensity attack bursts and periods of normal network activity. This variability is important for evaluating the robustness of the classifiers, as it forces them to distinguish between benign fluctuations in traffic and actual attack patterns. Additionally, the dataset is structured using fixed time windows, allowing for aggregated feature extraction and temporal analysis of packet behavior within each window. It is important to note that this dataset does not include other attack types, such as the earlier mentioned port scans. As a result, the evaluation focused specifically on the classifiers' ability to detect DoS activity and differentiate it from benign

traffic, while also enabling analysis of whether the models incorrectly classify traffic as other attack categories when such categories are absent. The following table summarizes key statistics of the dataset, including packet distribution, window composition, and traffic characteristics.

Table 5

Dataset statistics (DOS):

Metric	Value
Window size	0.05 seconds
Total packets	1,315,448
Total attack packets	996,838
Total benign packets	318,610
Total windows	1,068
Windows containing attack packets	208
Benign-only windows	860
Average total packets/window	1231.69
Average attack packets/window	933.37
Average benign packets/window	298.32
Avg attack packets (attack windows only)	4792.49
Avg benign packets (benign windows only)	27.59

The above table summarizes the statistical properties of the dataset containing DoS and benign traffic, which was used to evaluate the detection performance of the classifier models for this specific attack type. By isolating DoS activity within the dataset, the evaluation focused on the models' ability to accurately distinguish high-volume attack traffic from normal network behavior under controlled conditions. This setup enables a precise assessment of detection rate, false positives, and missed attacks without interference from other attack classes. Additionally, the inclusion of both attack-heavy and benign-only windows provided a realistic representation

of fluctuating network conditions, allowing the classifiers to be tested against varying traffic densities and temporal patterns. The table below lists the classifier performance metrics using this dataset.

Table 6

Classifier Performance Metrics (DoS attack):

Classifier	Accuracy	Errors	Precision (DoS=1)	Recall (DoS=1)	F1 Score (DoS=1)	ROC AUC
RTF	0.9999	1	0.9999	1.0000	0.9999	0.5007
MLP	0.9992	13	0.9990	0.9994	0.9992	0.5275
XGB	0.9999	2	0.9999	0.9999	0.9999	0.9365

The table above presents the performance of each classifier when evaluated on the DoS attack dataset. Across all models, the results demonstrate extremely high levels of accuracy, with values exceeding 99.9% in all cases. This indicates that each classifier is highly effective at distinguishing DoS attack traffic from benign traffic under the conditions represented in this dataset. The number of misclassifications (errors) remains very low for all models, with the Random Forest and XGBoost classifiers producing only 1 and 2 errors respectively, while the Multi-Layer Perceptron produced 13 errors. These low error counts further reinforce the reliability of the models in identifying attack patterns within the dataset. Precision and recall values for the DoS class (label = 1) are consistently near perfect across all classifiers. High precision indicates that when a classifier predicts a DoS attack, it is almost always correct, resulting in very few false positives. Similarly, the near-perfect recall values indicate that the models successfully detect nearly all DoS attack instances, with minimal missed attacks (false negatives). The resulting F1 scores, which balance precision and recall, also remain extremely

high, confirming the overall effectiveness of the classifiers. While the ROC AUC values vary between models, with XGBoost achieving the highest value, this metric is less informative in this specific evaluation due to the limited class diversity and highly separable nature of the dataset. The near-perfect classification performance suggests that the DoS attack patterns present in the dataset are highly distinguishable from benign traffic, allowing all models to perform exceptionally well. Overall, these results indicate that all three classifiers are capable of accurately detecting DoS attacks in this controlled setting, with only minor differences in performance between them. The table below provided a detailed summary of how each classifier distributed its predictions across the different output labels when evaluated on the DoS attack dataset. Rather than focusing solely on aggregate performance metrics, this table highlights the total number of instances each model classified as DoS, benign, or port scan, allowing for a more granular understanding of model behavior. This breakdown is particularly useful for identifying any bias in prediction tendencies, such as a model favoring one class over another, as well as verifying whether classifiers incorrectly assign samples to unrelated attack categories. In this case, the inclusion of port scan prediction columns enables analysis of cross-class misclassification, ensuring that no traffic is erroneously labeled as a different type of attack when such attacks are not present in the dataset. Overall, this summary complements the performance metrics by providing insight into how predictions are distributed across all possible classes.

Table 7

Prediction Distribution Summary (DoS attack):

Classifier	DoS predicted as 1	DoS predicted as 0	Portscan predicted as 1	Portscan predicted as 0	Benign predicted as 1	Benign predicted as 0	Total positive predictions across labels	Overall Accuracy
RTF	8017	8241	0	16258	8241	8017	16258	0.9999
MLP	8019	8239	0	16258	8239	8019	16258	0.9992
XGB	8016	8242	0	16258	8242	8016	16258	0.9999

The above table details the distribution of predictions produced by each classifier across all output labels when evaluated on the DoS attack dataset. This breakdown provided a comprehensive view of how each model assigns classifications, illustrating not only the number of correctly identified DoS and benign instances but also whether any predictions were incorrectly attributed to other attack categories. By examining the counts of predictions for each label, it becomes possible to assess the balance and consistency of each classifier's decision-making process. In particular, the table showed that all classifiers consistently distinguish between DoS and benign traffic, with prediction counts closely aligning with the ground truth distribution. The number of instances predicted as DoS (label = 1) and non-DoS (label = 0) reflected the models' ability to maintain accurate classification boundaries across the dataset. Furthermore, the absence of any predictions in the port scan category indicates that none of the classifiers incorrectly labeled traffic as a different type of attack, despite the availability of that output class. This confirms that the models do not exhibit cross-class confusion when evaluated on a dataset containing only DoS and benign traffic. Overall, this distribution-based analysis complements the performance metrics by providing deeper insight into classifier behavior,

confirming that the high accuracy observed is supported by consistent and well-balanced prediction patterns across all classes. The following table is a summary of the confusion matrix.

Table 8

Confusion Matrix Summary (DoS vs Benign):

Classifier	True Positives (DoS detected)	False Negatives (missed DoS)	False Positives (false alarms)	True Negatives (correct benign)	Total Errors
RTF	8016	0	1	8241	1
MLP	8011	5	8	8234	13
XGB	8015	1	1	8241	2

The above table summarizes the confusion matrix results for each classifier when distinguishing between DoS and benign traffic. This representation provided a detailed breakdown of classification outcomes, including correctly identified attack instances (true positives), missed attacks (false negatives), incorrectly flagged benign traffic (false positives), and correctly identified benign instances (true negatives). Unlike aggregate performance metrics, the confusion matrix offers a clearer view of the types of errors made by each model and their practical implications in a network security context. From the results, it can be observed that all classifiers achieve a high number of true positives and true negatives, indicating strong performance in both detecting DoS attacks and correctly identifying benign traffic. The Random Forest (RTF) classifier demonstrates perfect recall with zero false negatives, meaning it successfully detects all DoS attack instances in the dataset. However, it produced a single false positive, indicating a minimal occurrence of benign traffic being misclassified as an attack.

Similarly, the XGBoost (XGB) classifier maintains near-perfect performance with only one false negative and one false positive, reflecting a balanced trade-off between detection capability and false alarm rate. The Multi-Layer Perceptron (MLP), while still achieving very high performance overall, exhibits a slightly higher number of errors, including both false positives and false negatives. This indicates that, compared to the other models, the MLP is marginally more prone to both missing attack instances and incorrectly flagging benign traffic. Overall, the confusion matrix analysis confirms that all classifiers perform exceptionally well in this controlled DoS detection scenario, with only minor differences in error distribution. Importantly, the extremely low number of false negatives across all models highlights their effectiveness in minimizing missed attacks, which is a critical requirement for intrusion detection systems.

4.2.3 Port scan Attack Data Statistics

This section details the data results pertaining to the port scan attack packet capture. This dataset contained only port scan attack data where attacks are performed at random intervals. The below table details the dataset statistics of the port scan packet capture.

Table 9

Dataset Statistics (Portscan):

Metric	Value
Window size	0.05 seconds
Total packets	44,759
Total attack packets	8,032
Total benign packets	36,727
Total windows	1,774
Windows containing attack packets	15
Benign-only windows	1,759
Average total packets/window	25.23
Average attack packets/window	4.53
Average benign packets/window	20.70

Avg attack packets (attack windows only)	535.47
Avg benign packets (benign windows only)	16.17

This dataset consists of both port scan attack traffic and benign traffic, with attack events occurring within a relatively small number of time windows compared to the overall dataset. As shown, only a limited subset of windows contains attack packets, reflecting the typically sparse and distributed nature of port scan behavior, where probes are often spread across time and ports rather than concentrated in high-volume bursts. In contrast to the DoS dataset, which is characterized by high packet density during attack periods, the port scan dataset exhibits significantly lower packet counts per window, with attack traffic appearing more intermittently. This creates a more challenging detection scenario, as the distinction between benign and malicious activity is less pronounced. The relatively small number of attack windows, combined with the lower average number of attack packets per window, required classifiers to rely on more subtle patterns in traffic behavior rather than simple volume-based indicators. Overall, these statistics highlight the differences in traffic characteristics between port scan and DoS attacks and provide important context for interpreting classifier performance in subsequent evaluations.

Table 10

Classification Comparison (Portscan):

Classifier	Accuracy	Errors	Precision (Portscan=1)	Recall (Portscan=1)	F1 Score (Portscan=1)	ROC AUC
RTF	0.9945	10	1.0000	0.3750	0.5455	0.9997
MLP	0.9911	16	0.0000	0.0000	0.0000	0.0611
XGB	0.9956	8	1.0000	0.5000	0.6667	0.9991

The results demonstrate clear differences in classifier performance when evaluated on a port scan only dataset. Both the Random Forest Tree and XGBoost models achieved high overall

accuracy (0.9945 and 0.9956, respectively) and near-perfect ROC AUC values (≈0.999), indicating strong capability in ranking instances by likelihood of being an attack. However, this performance is misleading when considering detection effectiveness, as both models exhibit relatively low recall (0.375 for RTF and 0.500 for XGB), meaning a substantial portion of port scan attacks were not identified. In contrast, the Multi-Layer Perceptron (MLP) model performed poorly, failing to detect any port scan attacks, resulting in zero recall and F1-score, along with a very low ROC AUC (0.0611), indicating ineffective learning and poor probability ranking. Overall, these findings emphasize the importance of using multiple evaluation metrics, particularly recall and F1-score, in intrusion detection systems where missing attacks carries significant risk.

Table 11

Classifier detection results (Portscan):

Classifier	DoS predicted as 1	DoS predicted as 0	Portscan predicted as 1	Portscan predicted as 0	Benign predicted as 1	Benign predicted as 0	Total positive predictions across labels	Overall Accuracy
RTF	3	7638	183	7458	7455	186	7641	0.9757
MLP	53	7588	166	7475	7422	219	7641	0.9713
XGB	5	7636	181	7460	7455	186	7641	0.9757

The above table details the distribution of predictions produced by each classifier across all output labels when evaluated within the system architecture. This summary provided insight into how each model allocates its predictions between DoS, port scan, and benign classes, allowing for a more granular analysis of classifier behavior beyond aggregate performance metrics. From the results, it can be observed that the majority of predictions are assigned to the

benign class, which is consistent with the underlying distribution of the dataset. However, unlike the DoS evaluation, the classifiers exhibit a tendency to misclassify traffic into incorrect categories. In particular, a noticeable number of instances are predicted as port scan attacks despite the classifiers previously demonstrating difficulty in correctly identifying true port scan behavior. This suggests that while the models may occasionally assign the port scan label, these predictions are not aligned with actual attack instances. Additionally, the relatively small number of DoS predictions compared to benign predictions reflected the sparse nature of attack events within the dataset. The differences between classifiers are also apparent, with the MLP model producing a higher number of misclassifications compared to the Random Forest and XGBoost models, as indicated by its lower overall accuracy. In contrast, RTF and XGB demonstrate more consistent prediction patterns, with slightly higher accuracy and fewer incorrect classifications. Overall, this table highlights how each classifier behaves when deployed within the full architecture, revealing not only their strengths in identifying benign traffic but also their limitations in consistently detecting and correctly labeling less prominent attack types such as port scans. The below table showed the confusion matrix for the classifiers.

Table 12

Confusion Matrix Summary (Portscan vs Benign):

Classifier	True Positives (Portscan detected)	False Negatives (missed Portscan)	False Positives (false alarms)	True Negatives (correct benign)	Total Errors
RTF	180	5	3	7453	8
MLP	166	19	0	7456	19
XGB	181	4	0	7456	4

4.2.4 Live Detection Results

This section presents the live detection performance of the classification models during real-time inference. Latency is a critical metric in evaluating the practicality of the proposed architecture, as it directly impacts the system's ability to respond to threats in a timely manner. Consequently, multiple approaches were explored to minimize latency while maintaining detection effectiveness. It is important to note that reducing latency may introduce trade-offs, including an increased risk of false positives or false negatives due to reduced computational analysis. Three different inference strategies were evaluated in this study. All experiments utilized a window size of 0.05 seconds. This window size was selected because it represents the smallest feasible interval that allows multiple packets to be processed per window without resulting in empty windows or windows containing only a single packet. Empirically, this configuration ensured that each window contained approximately ten or more packets under typical traffic conditions. Live test results were obtained by executing the detection layer of the architecture while streaming benign network traffic. A Python-based logging script was used to record timing metrics, including latency and processing time. Latency was measured using three primary metrics: latency_ms, which represents the difference between the current system time and the timestamp of the most recent packet within a window; delay_ms, which represents the difference between the current time and the theoretical end of the 0.05-second window, and proc_ms, which captures the time required for feature extraction, feature reconstruction, and model inference. Among these, latency_ms served as the most direct representation of system latency. For example, a latency of 6000 ms indicates that the most recent packet in the window is six seconds old at the time of processing making latency six seconds. The first and most computationally intensive approach involved running all classifiers RTF, XGB, and MLP on

every window. While this method provided the most comprehensive and robust detection capability, it resulted in the highest latency.

Table 13

Inference Method Latency (Normal traffic)

Method	Latency_ms	Delay_ms	Proc_ms
All classifiers	6441.9	6440.7	164.1
RTF only inference	4780	4740	176
RTF every 5th packet	665	540	73

As shown in Table 13, this approach produced an average latency of 6441.9 ms, with a corresponding delay of 6440.7 ms and processing time of 164.1 ms. The second approach reduced computational overhead by running only the Random Forest classifier on each window, while reserving the other classifiers for potential future use. This resulted in a moderate reduction in latency. The average latency for this method was 4780 ms, with a delay of 4740 ms and a processing time of 176 ms, as shown in Table 13. Although this represents an improvement over the full ensemble approach, the reduction in latency is relatively limited. The third and final approach achieved the lowest latency by incorporating a selective sampling strategy. In this method, the Random Forest classifier is applied to incoming windows, and if traffic from a given source IP is classified as benign, subsequent packets from that source are sampled at a reduced rate (specifically, every fifth packet). These sampled packets are excluded from inference, effectively reducing the computational load. However, if malicious activity is detected, all three classifiers are executed on the relevant window to ensure accurate classification. This adaptive approach significantly reduces both processing time and overall latency. As shown in Table 13, this method achieved an average latency of 665 ms, a delay of 540 ms, and a processing time of 73 ms. Overall, the results demonstrate that the selective sampling strategy enables near real-

time detection performance while maintaining the ability to escalate to full ensemble analysis when necessary. This approach provided the most practical balance between latency and detection capability within the proposed architecture.

4.3 LLM Results and Response

This section presents the results of the LLM-generated incident summaries and the corresponding mitigation actions selected in response to detected network threats. To improve clarity, the responses are organized by attack scenario rather than by architecture. This structure allows the multi-agent and single-agent outputs to be compared directly for the same event, making differences in interpretation, mitigation strategy, and operational usefulness easier to evaluate.

4.3.1 Evaluation Approach

For each detected threat, both architectures were assessed using the same event context. The comparison focused on four factors: the correctness of the attack interpretation, the technical validity of the mitigation command, the operational usefulness of the response, and the extent to which the recommended action reflected the assumed deployment environment. The normalized outputs are summarized below so that the discussion emphasizes comparative analysis rather than repeating raw incident reports in full.

4.3.2 DoS Attack Response Comparison

4.3.2.1 Detected Event Overview

The detected DoS event involved traffic from 192.168.1.4 to 192.168.1.2. The event contained 342 packets, 308 observed source ports, and 36 observed destination ports. All three classifiers, Random Forest, XGBoost, and MLP, labeled the window as DoS, resulting in

unanimous agreement (3/3). These characteristics indicate a disruptive flooding pattern and provide a consistent basis for evaluating the two LLM-generated responses.

Table 14

Multi and Single-Agent DoS Comparison

Comparison Area	Multi-Agent	Single-Agent
Attack identification	Potential Low-Volume DoS	DoS Attack (High Confidence)
Key evidence emphasized	High packet volume and port diversity	Packet volume and port spread
Primary mitigation	FORWARD logging and drop rules	INPUT drop rule
Additional defensive depth	Includes event logging	Adds optional rate limiting
Main strength	Gateway-aware deployment	Mitigation sophistication
Main limitation	No rate limiting	Assumes host-based placement

4.3.2.2 Multi-Agent Response Analysis

The multi-agent system accurately interprets the DoS event and produced a mitigation strategy that is both valid and operationally coherent. Its response highlights the same indicators that support the classifier decision, namely high packet activity and broad port diversity, and the unanimous model agreement further strengthens confidence in the incident classification. A notable strength of this response is the layered mitigation design, which first logs the event and then blocks the traffic. The use of the FORWARD chain suggests that the recommendation is suitable for deployment on a network gateway or firewall, aligning well with routed network defense architectures.

```
sudo iptables -I FORWARD -s 192.168.1.4 -d 192.168.1.2 -j LOG \
    --log-prefix "BLOCKED DoS: " --log-level 4
sudo iptables -I FORWARD -s 192.168.1.4 -d 192.168.1.2 -j DROP
```

4.3.2.3 Single-Agent Response Analysis

The single-agent system also identifies the DoS attack correctly and presents a clear summary of the event. Its strongest contribution is the inclusion of an optional rate-limiting enhancement in addition to a direct blocking rule. This gives the response greater mitigation sophistication because it offers a mechanism for constraining abusive traffic while potentially preserving legitimate traffic patterns. However, the recommendation relies on the INPUT chain, which implies a host-based firewall perspective rather than a gateway-level deployment. As a result, the response is technically sound but somewhat less adaptive to network placement than the multi-agent output.

```
sudo iptables -I INPUT -s 192.168.1.4 -d 192.168.1.2 -j DROP

sudo iptables -A INPUT -s 192.168.1.4 -d 192.168.1.2 \
  -m hashlimit --hashlimit 10/s \
  --hashlimit-mode srcip --hashlimit-name dos_attack -j ACCEPT
sudo iptables -A INPUT -s 192.168.1.4 -d 192.168.1.2 -j DROP
```

4.3.2.4 Comparative Discussion

For the DoS case, both architectures produce accurate and immediately deployable responses. The multi-agent system is stronger in deployment awareness because its use of logging and the FORWARD chain better reflected a gateway-oriented defense model. The single-agent system, by contrast, is stronger in mitigation depth because it includes rate limiting in addition to direct blocking. Overall, the DoS comparison suggests that the multi-agent response is more context-aware from a network architecture perspective, whereas the single-agent response is more detailed in its defensive options.

4.3.3 Port Scan Response Comparison

4.3.3.1 Detected Event Overview

The port scan event involved traffic from 192.168.1.4 to 192.168.1.2 and was characterized by 38 packets, 15 observed source ports, and 23 observed destination ports. All three classifiers labeled the event as a port scan, again resulting in unanimous agreement (3/3). The relatively low packet volume combined with high destination port diversity is consistent with reconnaissance activity intended to identify accessible services prior to exploitation.

Table 15

Multi and Single-Agent Port scan Comparison

Comparison Area	Multi-Agent	Single-Agent
Attack identification	Confirmed Port Scan	Port Scan
Key evidence emphasized	Low-to-moderate packet volume with high destination port diversity	Systematic probing of multiple destination ports from a single source
Primary mitigation	INPUT-chain drop rule for the source and destination pair	INPUT-chain drop rule against the source IP
Additional guidance	Concise interpretation of reconnaissance behavior	Adds operational follow-up recommendations
Main strength	Clear and accurate classification of reconnaissance activity	Higher practical utility through follow-on guidance
Main limitation	No logging, rate limiting, or alerting included	Relies on a broad block rule rather than more nuanced handling

4.3.3.2 Multi-Agent Response Analysis

The multi-agent system correctly identifies the event as a port scan and explains the behavior in a way that aligns with recognized reconnaissance techniques. It emphasizes the combination of high destination port diversity and comparatively low packet count, which is appropriate for scan detection. The mitigation command is valid and immediately deployable,

but the response remains comparatively minimal. In contrast to its DoS handling, the port scan response does not include supporting measured such as logging or rate limiting, which reduces visibility into repeated probing.

```
iptables -I INPUT -s 192.168.1.4 -d 192.168.1.2 -j DROP
```

4.3.3.3 Single-Agent Response Analysis

The single-agent system also identifies the port scan accurately and provided a clear description of the reconnaissance behavior. Its main advantage in this case is that it extends beyond immediate containment by including operational recommendations to log repeated scan attempts, investigate the target host, and review firewall exposure. These follow-on recommendations increase the practical usefulness of the response because they support incident handling beyond a single blocking action. Nevertheless, the mitigation itself remains a broad source-based block and does not differentiate between short-lived scanning and more persistent abuse.

```
sudo iptables -I INPUT -s 192.168.1.4 -j DROP
```

Operational recommendations included by the single-agent system: log and monitor repeated scan attempts, investigate the target host for follow-on exploitation, and review firewall policies to restrict unnecessary open ports.

4.3.3.4 Comparative Discussion

For the port scan case, both architectures produce accurate summaries and technically valid firewall rules. The multi-agent output is concise and correct, but the single-agent output offers stronger practical value because it includes additional operational guidance beyond simple

blocking. As a result, the single-agent response is more useful for follow-up investigation, whereas the multi-agent response remains effective but narrower in scope.

4.3.4 Overall Comparative Findings

Across both attack scenarios, the multi-agent and single-agent architectures consistently identify the correct threat type and generate valid mitigation commands. Neither architecture is uniformly superior in every category. The multi-agent system demonstrates stronger contextual alignment with gateway-style deployment in the DoS case, particularly through its use of logging and the FORWARD chain. The single-agent system, however, provided more detailed defensive guidance, including rate limiting for DoS mitigation and follow-on operational recommendations for port scan response. Organizing the results by attack scenario makes this distinction clearer: the multi-agent architecture appears stronger in contextual framing, while the single-agent architecture appears stronger in mitigation depth and post-detection guidance.

4.3.5 LLM Live Response Performance

This section evaluates the performance of the LLM during live network detection. The model used in this study was a local GPT-OSS:20b, which is a relatively large model and therefore incurs substantial computational overhead during inference. As a result, the inclusion of the LLM layer significantly increases overall system latency when processing detected threats. During live testing, the latency associated with LLM-generated responses ranged from approximately 2 to 5 minutes following the detection of malicious traffic. This level of delay is expected, as large-scale LLMs require considerable processing time to generate detailed and context-aware outputs. While these responses provide valuable insights and mitigation strategies, they are not suitable for time-critical decision-making without additional optimization. Latency can be reduced by simplifying prompts, reducing context size, or limiting the number of LLM

calls. However, these optimizations introduce trade-offs, as shorter prompts may reduce the quality and completeness of the generated responses. In this context, the single-agent architecture demonstrates a notable advantage in terms of latency, as it required fewer LLM inference calls compared to a multi-agent system. Consequently, while multi-agent configurations may provide more comprehensive analysis, single-agent approaches offer improved responsiveness and are better suited for scenarios where low latency is a primary requirement.

Table 16

Multi-Agent vs Single-Agent latency

Agent	Time taken for final response
Multi-Agent	6 minutes 22 seconds
Single-Agent	5 minutes 34 seconds

From the table above we can see that both agents take a similar amount of time to generate a response against the attack. Once an attack is detected the architecture does not let packets through for the duration of time which an attack is detected and a response is being generated. When an attack is detected, since the LLM response generation will always be the slowest link the latency increases temporarily to, however long the LLM takes to form a response to the threat. The following tables demonstrate the LLM as a judge evaluation.

Table 17

Averaged LLM Evaluation Scores (Multi-Agent Response)

Criterion	Gemini	Claude	ChatGPT	Average
Accuracy	3	3	3	3
Completeness	3	2	2	2.33
Relevance	3	3	3	3.00
Actionability	3	2	2	2.33
Clarity	3	3	3	3.00
Hallucination	0	0	1	0.33

Overall score	3.00	2.60	2.40	2.67

Table 18

Averaged LLM Evaluation Scores (Single-Agent Response)

Criterion	Gemini	Claude	ChatGPT	Average
Accuracy	3	3	2	2.67
Completeness	3	2	3	2.67
Relevance	3	3	3	3.00
Actionability	3	2	2	2.33
Clarity	3	3	3	3.00
Hallucination	0	0	1	0.33
Overall score	3.00	2.60	2.60	2.73

Overall, the LLM-as-a-judge comparison indicates that both architectures produced responses of generally high quality, with only minor differences between them. The single-agent response achieved a slightly higher average overall score of 2.73, compared with 2.67 for the multi-agent response. Both approaches received identical average scores in relevance, actionability, clarity, and hallucination, suggesting that each system was similarly effective at producing readable, relevant, and largely grounded incident summaries. The primary difference emerged in accuracy and completeness. The multi-agent response achieved a perfect average accuracy score of 3.00, while the single-agent response scored slightly lower at 2.67. In contrast, the single-agent response performed better in completeness, with an average score of 2.67 compared with 2.33 for the multi-agent system. These results suggest that while the multi-agent architecture may provide slightly more precise threat characterization, the single-agent architecture produced marginally more complete overall responses. Taken together, the findings show that both systems are viable for AI-assisted incident response, though the single-agent approach demonstrated a small overall advantage in judged response quality.

4.3.6 Comparative Analysis of Single and Multiple Agent Response

This section provided a comparative evaluation of the multi-agent and single-agent systems in terms of detection accuracy, response quality, mitigation effectiveness, and overall system behavior. The goal is to identify strengths, weaknesses, and key differences between the two approaches when responding to both DoS and port scan attacks. From a detection standpoint, both architectures rely on the same underlying ensemble of classifiers (Random Forest, XGBoost, and MLP), and therefore exhibit identical detection capabilities. As such, there is no measurable difference in detection accuracy between the two systems, as both receive the same input features and classification outputs. The distinction between the architectures emerges primarily in how the detected threat information is processed and translated into actionable responses by the LLM layer. In terms of response quality, the multi-agent system consistently produced more structured, detailed, and context-aware outputs. By decomposing the task across multiple agents, the system is able to separately analyze the threat, retrieve relevant contextual information, and generate a refined response. This results in more comprehensive incident reports that include clearer justifications, better contextual grounding through the RAG pipeline, and more precise mitigation recommendations. In contrast, the single-agent system generates responses in a more direct manner, often producing shorter and less detailed outputs. While these responses are still correct and actionable, they may lack the depth of reasoning and contextual enrichment observed in the multi-agent approach. With respect to mitigation effectiveness, both systems are capable of generating valid defensive actions, such as firewall rules or packet filtering commands. However, the multi-agent system demonstrates a slight advantage in generating more contextually appropriate and adaptable mitigation strategies. This is due to its ability to incorporate retrieved knowledge and intermediate reasoning steps before producing a

final response. The single-agent system, while effective, tends to produce more generalized mitigation commands that may not be as finely tailored to the specific characteristics of the detected attack. In terms of system behavior and reliability, the multi-agent architecture introduces additional complexity through inter-agent communication and orchestration. While this enables improved reasoning and richer outputs, it also increases the likelihood of inconsistencies between intermediate agent outputs and contributes to higher computational overhead. The single-agent system, by contrast, offers a more streamlined and stable pipeline with fewer points of failure. This simplicity enhances reproducibility and makes the system easier to deploy and maintain in real-world environments. When considering latency, both systems are heavily constrained by the LLM inference time, as previously discussed. Although the multi-agent system required multiple LLM calls, the observed difference in total response time between the two architectures is relatively small. This suggests that the majority of latency is dominated by the base LLM processing time rather than the number of agent interactions. Nevertheless, the single-agent system maintains a slight advantage in responsiveness due to its reduced number of inference steps. Overall, the results indicate a clear trade-off between response quality and system efficiency. The multi-agent architecture excels in generating detailed, context-rich, and well-reasoned responses, making it suitable for scenarios where interpretability and comprehensive analysis are critical. Conversely, the single-agent architecture provided faster, simpler, and more efficient responses, making it better suited for real-time or resource-constrained environments. Future implementations may benefit from hybrid approaches that combine the efficiency of single-agent systems with the enhanced reasoning capabilities of multi-agent frameworks.

4.3.7 Detection Accuracy and Consistency

Both systems demonstrate strong detection capabilities across the evaluated attack scenarios. In cases where all three underlying classifiers (Random Forest, XGBoost, and MLP) agree, both the multi-agent and single-agent systems consistently produce correct classifications for both DoS and port scan attacks. However, differences emerge in ambiguous scenarios. The multi-agent system showed slightly improved interpretability by acknowledging mixed model outputs and incorporating this uncertainty into its reasoning. In contrast, the single-agent system tends to favor the ensemble majority without explicitly addressing disagreement between models. This can lead to overconfident classifications, particularly in cases where traffic patterns overlap between DoS and port scanning behavior. Overall, both systems achieve high detection accuracy, but the multi-agent system provided better transparency in decision-making, while the single-agent system demonstrates more decisive but less nuanced classification behavior.

4.3.8 Incident Summary Quality

The multi-agent system produced highly structured and consistent incident summaries, with clear separation between detection metrics, model consensus, and recommended actions. This structured output improves readability and aligns well with standard incident reporting formats used in security operations. In contrast, the single-agent system generates more detailed summaries, often including additional contextual explanations and operational recommendations. While this added detail enhances the depth of analysis, it can also introduce redundancy and reduce consistency across outputs. Thus, the multi-agent system excels in clarity and standardization, whereas the single-agent system provided greater descriptive depth at the cost of consistency.

4.3.9 Mitigation Strategy and Technical Sophistication

A key distinction between the two systems lies in the sophistication of the generated mitigation strategies. The multi-agent system primarily produced straightforward firewall rules that block malicious traffic. In some cases, it incorporates logging mechanisms, which provide additional visibility into attack activity. This results in simple, reliable, and immediately deployable mitigation actions. The single-agent system, however, demonstrates a more advanced understanding of defensive techniques by incorporating optional enhancements such as rate limiting using iptables modules (e.g., hashlimit). This reflected a more nuanced approach to mitigation, where traffic can be controlled rather than entirely blocked. Despite this advantage, both systems rely heavily on static blocking rules and do not dynamically adapt their mitigation strategies based on attack type. For example, port scanning activity is treated similarly to DoS attacks, even though reconnaissance behavior may be better handled through monitoring or throttling rather than immediate blocking.

4.3.10 Command Accuracy and Environmental Awareness

Both systems generate syntactically correct iptables commands, indicating a strong baseline understanding of firewall rule construction. However, neither system consistently accounts for the deployment environment. The multi-agent system sometimes used the FORWARD chain, suggesting an assumption of a gateway or router-based firewall, while the single-agent system predominantly used the INPUT chain, implying host-based deployment. Neither system explicitly verifies or adapts to the actual network context. The single-agent system partially addresses this limitation by occasionally suggesting alternative chains (e.g., FORWARD), demonstrating slightly greater awareness. However, both systems would benefit

from incorporating explicit environmental reasoning to ensure that generated rules are always applicable.

4.3.11 Overall Evaluation

The comparative analysis reveals that both systems are effective in detecting and responding to network-based threats, but they exhibit different strengths. The multi-agent system excels in structured reasoning, consistency, and clarity of output. It is better suited for environments where standardized reporting and interpretability are critical. The single-agent system provided more sophisticated and detailed mitigation strategies, including advanced techniques such as rate limiting, but lacks consistency and may produce less controlled outputs. In summary, the multi-agent approach offers better organization and explainability, while the single-agent approach provided greater technical depth in mitigation. A hybrid approach that combines the structured reasoning of multi-agent systems with the advanced mitigation capabilities of single-agent models could provide a more robust and effective solution for automated intrusion response.

CHAPTER 5. DISCUSSION

From the results presented in Chapter 4, classifiers such as Random Forest, XGBoost, and Multilayer Perceptron demonstrate strong capability in detecting network attacks and accurately classifying traffic. These models can be effectively integrated into both single-agent and multi-agent systems to generate structured summaries of detected incidents, along with corresponding mitigation strategies and recommended commands for blocking malicious activity. Chapter 5 evaluates the relative efficiency, interpretive quality, and overall effectiveness of the single-agent and multi-agent outputs in intrusion response.

5.1 RQ1

The experimental results indicate that the proposed multi-agent architecture is feasible as an intrusion analysis and response-support framework, but they only partially support HO1 when strict real-time end-to-end response was used as the standard. The evidence strongly supported the architecture's contextual reasoning and operational stability. In the multi-agent LLM-as-a-judge evaluation, the response achieved average scores of 3.00 for accuracy, 3.00 for relevance, and 3.00 for clarity, with an overall average score of 2.67/3 and a low hallucination score of 0.33. These results show that the architecture can reliably interpret classifier outputs and generate coherent incident summaries with useful mitigation guidance. The detection layer also demonstrated that near-real-time attack recognition is possible, although performance depends on the selected inference strategy. Under normal traffic, latency ranged from 665 ms using the RTF every-5th-packet method to 6441.9 ms when all classifiers were executed together, showing that detection can be relatively fast under lighter configurations but becomes slower under full ensemble inference. However, the final multi-agent response required 6 minutes and 22 seconds, which is substantially longer than would typically be considered real-time for operational

intrusion response. Therefore, the results support the feasibility of the multi-agent framework in terms of contextual reasoning, stability, and end-to-end functionality, but they do not fully satisfy the response-speed requirement for strict real-time intrusion response. HO1 is therefore partially supported, while the null hypothesis is rejected only with respect to contextual reasoning and operational stability, not end-to-end response speed.

5.2 RQ2

The results provide only partial support for HO2 and do not show that the multi-agent architecture outperformed the single-agent baseline across detection accuracy, response time, and overall response quality. At the detection level, performance is driven primarily by the shared classifier layer rather than by the agent architecture itself. The classifiers demonstrated strong detection capability, particularly for the port-scan dataset, where XGBoost achieved an accuracy of 0.9995, precision of 1.0000, recall of 0.9784, F1-score of 0.9891, and ROC AUC of 0.8489. However, because both the single-agent and multi-agent pipelines consume the same detection outputs, these results do not demonstrate that the multi-agent design provided higher detection accuracy than the baseline. The response-time results also do not support HO2. The multi-agent system required 6 minutes and 22 seconds to produce its final response, whereas the single-agent system completed its response in 5 minutes and 34 seconds, making the single-agent configuration faster in the evaluated scenario. The LLM-as-a-judge results similarly show that both architectures generated generally high-quality outputs, but the single-agent system achieved a slightly higher overall average score of 2.73/3 compared with 2.67/3 for the multi-agent system. The multi-agent response was judged slightly stronger in accuracy, scoring 3.00 versus 2.67, while both systems were equal in relevance, actionability, clarity, and hallucination control, and the single-agent response was judged more complete. Taken together, these findings suggest

that the primary advantage of the multi-agent architecture lies in its structured reasoning process and its ability to incorporate retrieved contextual knowledge when classifier outputs are ambiguous, rather than in measurable gains in detection accuracy or response speed. Therefore, HO2 is only partially supported in terms of adaptability and interpretive depth, while its claims of higher detection accuracy and faster response times are not supported by the current results. The null hypothesis cannot be fully rejected and remains valid for the speed and accuracy components of RQ2.

5.3 Limitations

This section outlines the primary limitations of the proposed architecture. One of the most significant limitations arises from the LLM layer. Both the single-agent and multi-agent configurations introduce additional latency during attack scenarios, as the time required for the LLM to generate a response is directly added to the overall system latency. While the LLM enhances contextual understanding and response generation, its computational overhead can negatively impact real-time performance, particularly in time-sensitive environments. Another limitation is associated with the detection layer, where each inference strategy presents inherent trade-offs. Utilizing all three classifiers (Random Forest, XGBoost, and MLP) in an ensemble improves detection robustness and accuracy; however, it significantly increases latency due to the added computational complexity. Conversely, relying solely on the Random Forest classifier as a fast-path detection mechanism reduces latency but increases the risk of undetected attacks, as a single model may not capture all attack patterns effectively. The adaptive sampling approach, which selectively skips packets following benign classifications, introduces additional vulnerabilities. While this method substantially reduces computational overhead and latency, it creates an opportunity for adversaries to exploit the sampling behavior. Specifically, an attacker

could establish a benign connection and subsequently launch malicious activity during periods when packets are not being analyzed. Although this risk is partially mitigated by periodic re-evaluation of the source through the RTF classifier, a sophisticated attacker could potentially infer the sampling pattern and time their activity to avoid detection within these unmonitored intervals. This highlights a fundamental trade-off between efficiency and security within the proposed system.

CHAPTER 6. CONCLUSIONS AND FUTURE WORK

This study has demonstrated that both single-agent and multi-agent intrusion response systems are capable of generating meaningful results and structured incident summaries when supported by machine learning classifiers for attack detection and traffic classification. The integration of classifiers such as Random Forest, XGBoost, and MLP provided a reliable foundation for identifying malicious activity within network traffic, while the use of large language models enables the transformation of these detections into human-readable summaries and actionable mitigation strategies. The findings indicate that both architectures are viable for intrusion response; however, the multi-agent approach offers enhanced interpretive capabilities, particularly in scenarios involving classifier disagreement or ambiguous attack patterns, by incorporating contextual reasoning and external knowledge retrieval. Despite these strengths, several limitations were identified that present opportunities for future research and system improvement. One key area for further investigation is the scalability and generalization capability of the classifiers when trained on larger and more diverse datasets. While the current models perform well within the evaluated scenarios, their effectiveness may be impacted by increased variability in attack types, traffic patterns, and real-world network conditions. Future work should therefore focus on expanding the training datasets to include a broader range of attack vectors and evaluating the limits of classifier performance in terms of detection accuracy, class imbalance, and robustness to previously unseen threats. Another important direction for future research involves enhancing the retrieval mechanisms used within the RAG component of the system. The current implementation demonstrates the value of incorporating external knowledge to support contextual reasoning; however, improvements in document indexing, ranking strategies, and semantic retrieval techniques could further increase the relevance and

accuracy of the information provided to the language model. Exploring alternative embedding models, hybrid retrieval methods, and adaptive context selection strategies may significantly improve the overall quality of generated responses. Additionally, the system could be extended to support the analysis of encrypted network traffic through the use of Transport Layer Security (TLS) key integration. By enabling controlled decryption of network flows, the system would gain visibility into payload-level data, allowing for more precise detection of sophisticated or obfuscated attacks that may not be identifiable through metadata alone. This capability would be particularly valuable in modern network environments where a large proportion of traffic is encrypted. Finally, a promising area for future development is the introduction of an orchestration mechanism that enables the system to adapt and update itself over time. Specifically, the integration of an orchestrator agent capable of monitoring newly identified threats within the RAG database and triggering retraining or fine-tuning of classifier models would allow the system to evolve in response to emerging attack patterns. Such a self-updating architecture would reduce reliance on manual intervention and improve long-term effectiveness, moving toward a more autonomous and adaptive intrusion response system. In conclusion, this research establishes a strong foundation for the use of large language models within both single-agent and multi-agent intrusion response frameworks. While the current system demonstrates feasibility and effectiveness in controlled environments, future enhancements in classifier scalability, retrieval mechanisms, encrypted traffic analysis, and autonomous model updating have the potential to significantly advance the capabilities of intelligent intrusion response systems.

REFERENCES

Adnan, M. T., Noor, S. B., Ghosh, S. K., Lee, J.-M., & Kim, D.-S. (2025). CoLLAID: Collaborative lightweight language model for adaptive intrusion detection in maritime IoT. 2025 International Conference on Mobile, Military, Maritime IT Convergence (ICMIC), 210–213. https://doi.org/10.1109/ICMIC66299.2025.11257800

Aquino, G. E. (2025). From RAG to multi-agent systems: A survey of modern approaches in LLM development. Preprints.org. https://www.preprints.org/manuscript/202502.0406/v1

Houssel, P. R. B., Singh, P., Layeghy, S., & Portmann, M. (2024). Towards explainable network intrusion detection using large language models. In Proceedings of the IEEE/ACM International Conference on Big Data Computing, Applications and Technologies (BDCAT) (pp. 67–76). IEEE. https://ieeexplore.ieee.org/document/10946944

Hu, Y., Cheng, Z., Li, X., & Wang, S. (2024). MySQL-Pot: A LLM-based honeypot for MySQL threat protection. In Proceedings of the IEEE International Conference on Big Data Analytics. https://ieeexplore.ieee.org/document/10607309

Hung, S.-L., Chen, C.-K., Furumoto, K., Takahashi, T., & Sun, H.-M. (2024). Dark Watchdog: A novel RAG-driven system for real-time detection and analysis of data leaks on dark web forums. 2024 Annual Computer Security Applications Conference Workshops (ACSAC Workshops), 11–18. IEEE. https://doi.org/10.1109/ACSACW65225.2024.00010

Ikbarieh, S., Gupta, M., & Mahalal, E. (2025). LLM-based multi-class attack analysis and mitigation framework in IoT/IIoT networks. In Proceedings of the IEEE Global Conference on Artificial Intelligence and Internet of Things (GCAIoT). IEEE. https://ieeexplore.ieee.org/document/11275542

Ismail, K., Kurnia, R., Brata, Z. A., Nelistiani, G. A., Heo, S., Kim, H., & Kim, H. (2025). Toward robust security orchestration and automated response in security operations centers with a hyper-automation approach using agentic artificial intelligence. Information, 16(5), 365. https://doi.org/10.3390/info16050365

Karunanayake, B., Khalil, I., Yi, X., & Lam, K.-Y. (2025). Toward LLM-driven adaptive policy orchestration for host-based intrusion detection systems in IoT environments. IEEE Network, 39(5), 66–73. https://doi.org/10.1109/MNET.2025.3579532

Kramer, B., Dietrich, S., & Lagesse, B. (2025). Integrating large language models into security incident response. In Proceedings of the Symposium on Usable Privacy and Security (SOUPS). USENIX Association. https://www.usenix.org/system/files/soups2025-kramer.pdf

Lee, Y.-H., Han, C., Peng, M.-C., & Takahashi, T. (2025). LLM-based multi-label mapping of Snort rules to ATT&CK. 2025 IEEE Conference on Dependable and Secure Computing (DSC). IEEE. https://doi.org/10.1109/DSC65356.2025.11260863

Li, J. T. M., & Yau, P. C. Y. (2025). A chatbot-assisted intrusion detection system for network traffic analysis using machine learning and retrieval-augmented generation. 2025 5th International Conference on Computer Systems (ICCS), 180–186. IEEE. https://doi.org/10.1109/ICCS67844.2025.11291869

Lin, X., Zhang, J., Deng, G., Liu, T., Liu, X., Yang, C., Zhang, T., Guo, Q., & Chen, R. (2025). IRCopilot: Automated incident response with large language models (arXiv:2505.20945). arXiv. https://arxiv.org/abs/2505.20945

Ma, Y., Zeng, G., Zhang, P., Huang, Y., Zhang, F., Lin, R., & Qian, H. (2025). TrapLLM: An LLM-powered interactive log-based honeypot for real-world network attacks. 2025 IEEE 24th

International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), 1578–1585. IEEE. https://doi.org/10.1109/Trustcom66490.2025.00183

Manor, O., Lavi, O., Schwartz, T., Sekiya, M., Suga, J., Hikichi, K., Unno, Y., & Murillo, A. F. (2024). Retrieval of network packets information using a generated latent space representation for network analysis in cyber security. 2024 Annual Computer Security Applications Conference Workshops (ACSAC Workshops), 224–231. IEEE. https://doi.org/10.1109/ACSACW65225.2024.00032

Mei, L., Mo, S., Yang, Z., & Chen, C. (2025). A survey of multimodal retrieval-augmented generation (arXiv:2504.08748). arXiv. https://arxiv.org/pdf/2504.08748

Nguyen, T., Chin, P., & Tai, Y.-W. (2025). MA-RAG: Multi-agent retrieval-augmented generation via collaborative chain-of-thought reasoning (arXiv:2505.20096). arXiv. https://arxiv.org/pdf/2505.20096

Saeed, I. A., Selamat, A., Rohani, M. F., Krejcar, O., & Chaudhry, J. A. (2020). A systematic state-of-the-art analysis of multi-agent intrusion detection. IEEE Access, 8, 180184–180197. https://doi.org/10.1109/ACCESS.2020.3027463

Shan, R. (2024). Certifying generative AI: Retrieval-augmented generation chatbots in high-stakes environments. Computer, 57(9), 35–43. https://doi.org/10.1109/MC.2024.3401085

Sudasinghe, P. B., Liyanage, K. S. K., & Pussewalage, H. S. G. (2025). Lightweight LLMs for network attack detection in IoT networks. 2025 Computing, Communications and IoT Applications (ComComAp), 395–400. IEEE. https://doi.org/10.1109/ComComAp68359.2025.11353200

Sudheer, A., Chou, C.-H., & Kumar, S. (2025). Demo: A real-time multi-agent network attack detection and incident response system. In 2025 IEEE Symposium on Visual Cybersecurity (SVCC) (pp. 1–3). IEEE. https://doi.org/10.1109/SVCC65277.2025.11133641

Xu, Z., Wang, Y., Li, J., Chen, R., & Liu, T. (2024). Can LLMs beat humans in debating? A dynamic multi-agent framework for competitive debate (arXiv:2408.04472). arXiv. https://arxiv.org/pdf/2408.04472

Zhang, H., Bin Sediq, A., Afana, A., & Erol-Kantarci, M. (2024). Large language models in wireless application design: In-context learning-enhanced automatic network intrusion detection. In GLOBECOM 2024—2024 IEEE Global Communications Conference (pp. 2479–2484). IEEE. https://doi.org/10.1109/GLOBECOM52923.2024.10901312