Enhancing Network and Endpoint Detection and Response Systems: A Grounded LLM-Assisted Contextual Analysis

University of Turku
Department of Computing
Master of Science, Cybersecurity

June 2026

Tanuraj Saha

Supervisors:
Tahir Mohammad
Petri Sainio

UNIVERSITY OF TURKU
Department of Computing

Tanuraj Saha: Enhancing Network and Endpoint Detection and Response Systems: A Grounded LLM-Assisted Contextual Analysis

Master of Science, Cybersecurity, 85 p.

June 2026

Modern Security Operations Centres (SOCs) face challenges due to a large number of fragmented security alerts originating from both external and internal networks. Specifically, in the case of lateral movement of internal traffic, they originate from siloed Network Detection and Response (NDR) and Endpoint Detection and Response (EDR) technologies. To address these issues, this thesis presents an end-to-end multi-layered architecture for automated alert correlation and contextual reasoning. This thesis focuses primarily on the post-detection phase by designing a contextual framework that groups related alerts into structured incidents, rather than developing telemetry integration or detection algorithms. This contextual analysis of the Large Language Models (LLMs) can help generate short, evidence-based summaries and investigative recommendations. Earlier, analysts had to manually reconstruct threat narratives, which can often lead to alert fatigue, increased cognitive load, and delayed incident response when there is no structural aggregation. One of the major analytical capabilities of LLMs is the automation of triage for SOCs. However, their implementation in enterprise SOCs is often delayed by inadequate contextual data and strong data protection regulations.

The methodology involves several steps. First, a detection layer with machine learning models (such as XGBoost, random forest, and autoencoders) trained on public datasets. Then, a correlation layer integrates these network anomalies with endpoint logs, converting a large number of alerts into a limited number of incidents. The contextualization layer evaluates the reasoning capabilities of both cloud-based (Gemini 2.5 Flash, OpenAI gpt-4o) and locally deployed LLMs (Llama 3, Gemma 3). Cloud-based LLMs performed better at detailed interpretation. In summary, this thesis demonstrates that grounding LLMs in deterministically linked NDR and EDR telemetry substantially improves automated threat analysis. The system converts discrete alerts into standardised playbook-ready intelligence to support Tier-1 SOC triage. It provides a viable, privacy-preserving system for deploying sophisticated generative AI in secure enterprise environments. The machine learning detection layer achieved an F1-score close to 0.99. The deterministic correlation engine reduced telemetry to an Alert-to-Incident Ratio (AIR) of 95. Subsequently, the grounded LLM produced MITRE-aligned narratives with near-zero hallucinations, reducing triage time by up to 66 percent.

Keywords: Network detection and response, Endpoint detection and response, Large Language Model, Alert correlation, Contextual analysis, Cybersecurity, Security Operation Centre,SOC Triage

Contents

List of Figures

List of Tables

List of Acronyms

AI Artificial Intelligence

AIR Alert-to-Incident Ratio

API Application Programming Interface

APT Advanced Persistent Threat

AUC Area Under the Curve

CIC Canadian Institute for Cybersecurity

CTI Cyber Threat Intelligence

EDR Endpoint Detection and Response

IAT Inter-Arrival Time

IDS Intrusion Detection System

IF Isolation Forest

IoC Indicator of Compromise

JSON JavaScript Object Notation

LLM Large Language Model

ML Machine Learning

MSE Mean Squared Error

MTTD Mean Time to Detect

MTTR Mean Time to Respond

NDR Network Detection and Response

NIS2 Network and Information Security Directive 2

PCAP Packet Capture

RAG Retrieval-Augmented Generation

RDP Remote Desktop Protocol

RF Random Forest

SDN Software-Defined Networking

SIEM Security Information and Event Management

SMB Server Message Block

SOAR Security Orchestration, Automation, and Response

SOC Security Operations Center

SSH Secure Shell

TCP Transmission Control Protocol

WMI Windows Management Instrumentation

XAI Explainable Artificial Intelligence

XDR Extended Detection and Response

XGBoost Extreme Gradient Boosting

Statement of AI Usage

For preparing this master's thesis, Generative AI (Gemini) was employed as a support tool for searching, organising and structuring information. Google Scholar and ChatGPT were used to locate academic literature and track citations for the references. Grammarly was used for proofreading, which helped to refine the writing and correct any grammatical errors. In addition, generative AI (ChatGPT) was used to generate architectural drawings and conceptual diagrams to ensure clarity and persuasiveness. The SOC incident reports are also included as Appendix B and were totally auto-generated by an LLM.

All AI-assisted changes and suggestions were extensively reviewed to verify that they are accurate, original and academically legitimate. They were rigorously verified to align with the research goals and experimental outcomes. But the research concept, analysis, and experiments are all conducted by the author himself.

1 Introduction

Modern enterprise networks generate large volumes of telemetry data, including network flows, authentication logs, intrusion-detection alerts, and endpoint events across various systems and applications. In the past, network security monitoring relied primarily on packet capture (PCAP) and flow-based telemetry, such as NetFlow, to identify anomalous traffic patterns [1]. These methods previously worked well for signature-based and rule-driven inspection, but they required extensive manual correlation. They did not show what was happening at the host level, especially for behavioural analysis. As enterprise environments became more complex and encrypted, traffic-based monitoring was insufficient to detect advanced threats such as lateral movement and credential exploitation. This evolution introduced the Network Detection and Response (NDR) and Endpoint Detection and Response (EDR) for host-based telemetry data [2]. Increasingly, they are being utilised for various security tasks, particularly for log analysis, alert triage, and incident summary. Modern NDR and EDR platforms use ML and AI to detect unusual behaviour, reduce false positives, and improve automated threat prioritisation in large-scale environments. NDR and EDR solutions are widely used to detect threats, including malware infections, lateral movement, credential abuse, and insider activity [3,4]. ML models that use structured features from this kind of telemetry can perform well on benchmark datasets. However, these models typically provide analysts only with anomaly scores or classification labels, which offer little context. Detection

outputs are often fragmented and difficult to interpret, so analysts must manually link and correlate them. Structured contextual correlation can provide large language models (LLMs) with focused, evidence-based information to summarise alerts, support triage, and enable SOC security automation. However, the operational efficiency of these models is explicitly dependent on proper contextual analysis [5]. Without structured and rigid evidence-based data, LLMs are very prone to produce hallucinations. This can lead to irrelevant or factually incorrect threat intelligence within the investigation workflow [6].

As modern network infrastructure expands, attackers can move laterally more easily. Attackers can carry out credential-based attacks, although they appear to be legitimate administrative tasks. While machine learning can detect unusual behaviour, it often produces too many false positives and alerts. This adds extra hurdles for SOC analysts and results in slower response times. As a result, security teams closely monitor metrics such as Mean Time to Detect (MTTD) and Mean Time to Respond (MTTR). This shows that not only is it essential to find threats quickly, but also to make them easier to understand and sort through [7].

The main focus of this thesis is to enhance the post-detection phase of ML-based detection systems using structured correlation. However, the grounded LLM helps to analyse the incidents with a contextualised summary.

1.1 Problem Statement

Nowadays, many IT organizations use NDR and EDR systems to detect and monitor internal traffic. However, lateral movement and credential-based attacks can be challenging to detect and investigate. ML-based detectors are useful for detecting anomalous traffic, but sometimes generate fragmented alerts with no explanation. Consequently, security analysts have to manually correlate similar events, which slows down investigations and responses [8]. Most current research has focused on improving detection accuracy or on exploring LLM-based summarisation algorithms [40,41]. This thesis attempts to fill this gap by exploring methods to increase the performance of these detection systems through contextual correlation and evidence-bound LLMs. The main focus of this thesis is to improve the post-detection framework for NDR and EDR systems [7,22]. This thesis studies an architecture where:

- Anomalies and alerts are generated based on their severity using the built-in ML models used by the NDR or EDR. Structured ML models, such as Random Forests, provide alerts and detection scores.
- Predefined rules (e.g., time periods, shared entities) assist in grouping alerts into logical incident bundles.
- A grounded LLM is integrated to explain the incident based on evidence without changing the original verdicts provided by these detection tools to identify the specific problem.

The primary objective is to make structured ML detection models more understandable, more successful in the analysis of data, and more effective for operations.

1.2 Research Questions

To address the challenges of alert fatigue and high cognitive load on SOC analysts, this thesis formulates the following research questions:

1. What are the key operational and interpretability constraints in the case of NDR and EDR for lateral movement contexts and insider attacks?
2. How does contextually structured correlation reduce fragmented alerts, improve analysis clarity, and lower the hallucination rate?
3. How well can tuning the ML-based detection and fusion of alerts find a balance between lowering false positives and increasing sensitivity?
4. How does grounded LLM-assisted contextual reasoning affect metrics of operational efficiency like MTTD, MTTR, and analyst workload when compared to detection outputs?

1.3 Research Objectives

The goal of this thesis is to investigate how structured contextual correlation and grounded LLM-assisted reasoning can improve the performance of ML-based NDR and EDR systems in detecting lateral movement. More specifically, the objectives of this thesis are:

1. To identify the operational and interpretability limitations for the existing NDR and EDR systems for lateral movement and insider threats.
2. To develop a structured correlation mechanism which connects multiple fragmented alerts to enhance investigation clarity.
3. To optimise the performance of ML models to achieve a strict balance between reducing false positives in terms of detection and maintaining threat sensitivity.

4. To assess the effectiveness of grounded LLM-assisted reasoning on operational efficiency metrics such as analyst workload, MTTD, and MTTR.

1.4 Research Contributions

This thesis mainly focuses on the enhancement of the operational efficiency of NDR and EDR systems through the following contributions:

1. Presenting a Security Operations Centre (SOC) end-to-end pipeline which integrates machine learning detection, alert correlation, and contextual analysis supported by LLMs.

2. The proposed correlation system aggregates related alerts generated from NDR and EDR into incident bundles based on time and entity relationships.

3. Evaluation of various machine learning detection models, such as Random Forest, XGBoost, Isolation Forest, and Autoencoder, to identify anomalies in network telemetry.

4. A grounded LLM layer is incorporated to convert structured incident data into contextual narratives that analysts can interpret.

5. An experimental validation demonstrates that contextual reasoning can improve SOC analysts' efficiency in understanding correlated incidents.

1.5 Thesis Structure

The remainder of this thesis is organised as follows: Chapter 2 presents an organised and relevant review of modern cyber threats, lateral movement, NDR, EDR, machine-learning techniques for intrusion detection, and recent advances in contextual analysis and large language models for security operations.

Chapter 3 describes the proposed methodology, including the overall system architecture, integration of NDR and EDR telemetry, feature extraction, machine-learning detection pipelines, and design of the contextual analysis and LLM-assisted interpretation layer.

Chapter 4 details the experimental setup, including the use of publicly available network- and host-based datasets, data preprocessing steps, model training procedures, and a step-by-step experimental workflow for evaluating the detection and response performance.

Chapter 5 presents the evaluation results, focusing on detection accuracy metrics (precision, recall, and F1-score) and operational metrics (MTTD and MTTR), and analyses the impact of NDR–EDR integration and LLM-assisted contextual analysis.

Chapter 6 concludes the thesis by summarising the main contributions, the study's limitations, the implications for practical deployment and regulatory compliance, and directions for future research.

2 Literature Review

This chapter provides an overview of key concepts and relevant research for the thesis. First, the mechanisms of lateral movement and internal network threats are discussed in this chapter. Next, the development, capabilities, and challenges of NDR and EDR are addressed. The chapter also examines the application of ML and LLMs in cybersecurity. It also emphasizes both advantages and limitations faced by SOCs. Finally, it summarises recent advancements and research in the field and identifies the research gaps that the proposed framework aims to reduce.

2.1 Lateral Movement and Internal Threats

Lateral movement refers to the methods an attacker applies after gaining initial access to a system. Attackers use this access to traverse connected systems, accounts, and internal resources within the network infrastructure. The purpose is to identify critical resources such as database servers, domain controllers, or hosts running sensitive applications. Lateral movement primarily occurs within the internal network, whereas perimeter attacks occur at the network edge. Unlike perimeter attacks, which primarily target the network edge, lateral movement leverages the trusted internal environment. These attacks can be mitigated in some contexts by employing modern access controls such as Zero-Trust Architectures and microsegmentation via Software-Defined Networking (SDN). These technologies can only be used to isolate the internal network. However, detecting actual lateral movement remains a major

challenge despite these structural defences. These challenges are described in detail in Section 2.2. Because threat actors often employ "living-off-the-land" strategies, their activities are difficult to distinguish from legitimate administrative network traffic [9,10].

Attackers often employ remote administration tools (command-and-control attacks) and protocols to move laterally. They typically use attack tools such as Windows Remote Management (WinRM), Server Message Block (SMB), and Remote Desktop Protocol (RDP). Attackers can use tools such as PsExec to run commands on other machines over SMB by creating temporary services. They also use Windows Management Instrumentation (WMI) to run commands remotely without installing any additional software. Attackers can also use stolen credentials to log in to other systems [10,11]. Although tools like SMB, RDP, and WMI are specific to Windows, lateral movement via SSH or Cron jobs is a risk for all operating systems, including Linux and macOS. Services such as Secure Shell (SSH), Network File System (NFS), Virtual Network Computing (VNC), and Apple Remote Desktop (ARD) pose similar risks for lateral movement within a network. This thesis focuses on Windows tools as examples because most enterprise networks, especially in enterprise desktop environments, rely on Windows. Still, the detection, correlation, and contextualisation methods discussed here are platform-agnostic and can be used across different systems within an enterprise network.

A key challenge in detecting lateral movement is that these activities closely resemble routine IT administration and troubleshooting. Isolated incidents, such as remote logins or service creation, are rarely malicious. Effective detection, therefore, requires correlating multiple weak signals across time, hosts, and users, and distinguishing unexpected behaviour from baseline administrative patterns [12,13].

2.2 Network Detection and Response (NDR)

NDR systems monitor network traffic for anomalous behaviour that could indicate a security breach. Modern NDR solutions, unlike traditional systems that relied on known attack patterns, service usage, major behaviour changes and internal traffic to obtain a clearer picture of the situation. In fact, NDR detects anomalies if an attacker attempts to use an incorrect or non-standard protocol on a particular port [9,75].

NDR collects important information, including how often connections are established, the authentication mechanism, how services are used, and how to handle major changes in behaviour. This information helps find unusual network activity or unexpected connections between devices. Although NDR offers many benefits, it also introduces operational challenges. Encryption is now widely used, thereby hiding payload data and making it harder to profile applications or analyse traffic in detail [14,15]. Network configurations such as asymmetric routing and NAT also make it difficult to accurately identify endpoints and assets. NDR sometimes faces challenges with SDN, particularly in lateral communication and microsegmentation. This design reduces the attack surface by hiding targets from attackers, but it also makes it more difficult for defenders to observe events. East-west traffic often remains within virtual segments, thereby avoiding centralised monitoring. Overlay networking protocols such as VXLAN and Geneve add another layer of complexity, as their encapsulated traffic is much harder to inspect in depth. Moreover, SDN is facilitated by the OpenFlow protocol. Because SDN employs dynamic routing, it often bypasses traditional static NDR sensors, resulting in visibility gaps [15,16].

Because encryption is so common, NDRs often rely on flow-level metadata. An anomalous connection does not necessarily indicate that a device has been hacked or compromised; additional information from the destination is needed. As a result, NDR alerts may not always provide sufficient evidence to justify action [17]. NDR

systems analyse network traffic and metadata to identify unusual behaviour that may indicate a breach or an attacker. Modern NDR focuses on behavioural analysis, identifying suspicious events, and internal (east-west) communications, with a strong emphasis on context. This strategy differs from traditional intrusion detection systems that rely on signatures [18]. It is especially important to identify lateral movement, which manifests as unusual internal connections, scanning activity, or the use of the wrong protocol; the number of times users connect; how they log in; how they use services; and how these patterns differ from previous ones. This openness makes it easier to spot strange internal communications, unexpected peer-to-peer exchanges, or questionable use of administrative processes. Thus, NDR alerts alone may not be sufficient to inform decisions about event management [19].

2.3 Endpoint Detection and Response (EDR)

The primary function of EDR systems is to monitor internal host activity to detect and respond to potential security breaches. They collect detailed telemetry data from endpoints, including process creation, service installation, registry changes, file changes, and user authentication [20,68]. SOC analysts can utilise this information to identify unusual traffic patterns that may indicate malware, exploitation, or unauthorised access.

However, EDR has some limitations as well:

- EDR agents may not cover unmanaged or offline devices, resulting in coverage gaps.
- They primarily focus on host behaviour and may lack sufficient network-level information, making it difficult to track lateral movement or coordinated attacks across multiple endpoints [21].

For example, if an attacker compromises a workstation and then accesses a file server, the EDR would detect anomalous activity on both computers. Still, it may lack sufficient meaning without correlation or network-based evidence.

2.4 Extended Detection and Response (XDR)

Extended Detection and Response (XDR) provides broader visibility than standalone NDR and EDR systems by aggregating telemetry across network, endpoint, and cloud domains [86]. It leverages cross-domain machine learning to automatically correlate fragmented indicators of compromise (IoCs) into unified incident graphs. XDR platforms usually collect data from various sources and automatically group alerts. This thesis does not compete with commercial XDR tools. Instead, the proposed framework works with XDR by focusing on what happens after threats are detected. Here, structured NDR–EDR incident summaries are correlated in the same time window and turned into clear, evidence-based incident summaries using contextual reasoning, providing analysts with the information they need to decide and act.

2.5 Security Orchestration, Automation and Response (SOAR)

Security Orchestration, Automation, and Response (SOAR) platforms use automated incident triage workflows. These are called "playbooks," which improve the efficiency of security operations. Security Information and Event Management (SIEM) detects anomalies; on the other hand, SOAR automatically queries external threat intelligence, extracts relevant logs, and initiates containment actions such as isolating a host [86]. This automation significantly reduces both the Mean Time to

Respond (MTTR) and the analyst workload. But SOAR is only as good as the quality of the alerts it receives; feeding fragmented ML detections into SOAR can create redundant workflows, leading to automation fatigue rather than operational efficiency.

2.6 Machine Learning for Anomaly Detection

Traditional and rule-based systems depend on human engineers, whereas machine learning algorithms can detect underlying patterns and correlations in datasets to inform decision-making. These models are trained on samples characterised by numerical or categorical data and adaptively adjust their internal parameters to categorise events, e.g., as 'regular' or 'malicious' network traffic. The model is trained on this logic and can generalise to correctly assess novel, unseen telemetry. After being trained, the model can apply this information to new data. This method works well when behaviour is too complicated or changes too quickly for set rules to work [22].

In intrusion detection, supervised learning is effective when each instance is categorised as either benign or malicious. Decision trees, random forests, and neural networks are examples of classifiers that learn to distinguish these groups. These models work well if the labels are correct and the environment is similar to the training data. However, they may struggle when the data changes or is attacked in novel ways, leading to poor generalisation and more false positives [23].

2.7 Application of AI/ML for Modern Network and Endpoint Security

AI and ML enhance network and endpoint security by detecting unusual activity, predicting threats, and enabling rapid automated responses. Unlike signature-

based solutions, these systems can learn normal behaviour from telemetry and detect anomalies such as malware, intrusion or lateral movement. Moreover, these technologies establish the baseline for the security team's regular activities and support the interpretation of alerts and incidents involving lateral movement in a more human-friendly context [65].

2.7.1 AI/ML in Network Detection and Response (NDR)

A large part of NDR systems relies on machine learning. It enables them to analyse large volumes of network data and identify behaviour that deviates from typical communication patterns. NDR systems examine flow-level and protocol-level metadata instead of encrypted packet payloads. This information includes connection frequency, session duration, data volume, protocol distribution, and traffic direction from east to west. By learning how networks typically operate, these models can detect anomalous behaviour related to reconnaissance, internal scanning, lateral movement, and command-and-control [23,24]. Machine learning is effective for identifying users who exploit lawful administrative protocols such as Windows Management Instrumentation (WMI), Server Message Block (SMB), Remote Desktop Protocol (RDP), and WinRM. Since these protocols are widely used in enterprise environments, static rule-based systems frequently misclassify malicious activity as legitimate network traffic [10]. Instead, NDR models look for small behavioural changes that indicate internal attacks, such as unusual connections, unusual access timings, or unexpected protocol sequences [26].

2.7.2 AI/ML in Endpoint Detection and Response (EDR)

In EDR systems, machine learning analyses operating system logs and telemetry data from endpoint agents. This telemetry includes creating new processes, establishing parent-child relationships, logging in and verifying identity, setting up ser-

vices and scheduled tasks, modifying the registry, and persisting artefacts [27]. EDR that uses machine learning to find out the difference between unusual behaviour and normal behaviour by learning how users and processes normally work on a host.

In many cases, supervised learning models are used to detect known but dangerous behaviour when labelled data are available. This helps place known attack behaviours in the correct category [28]. Unsupervised and semi-supervised methods, on the other hand, learn what normal endpoint activity looks like and identify changes that could indicate new attacks, anomalous behaviour, or insider exploitation. These methods can help protect against previously unknown threats but can also increase false positives, especially in settings with many concurrent administrative tasks [29].

2.7.3 AI/ML Pipelines with NDR–EDR Correlation

When NDR and EDR systems were used in isolation, each exhibited visibility gaps. NDR might detect abnormal network activity but not determine whether it occurs on endpoints. EDR might identify unusual behaviour on a host or endpoint but not determine how it propagates throughout the network. Recent studies have consequently focused on integrated AI/ML pipelines that leverage shared properties, such as hosts, users, IP addresses, and time frames, to integrate evidence from networks and endpoints. Integrated pipelines can reduce confusion and increase detection confidence by merging NDR and EDR characteristics or by linking their alarms. Network anomalies supported by endpoint data are more likely to be real attacks. In contrast, isolated endpoint alarms are meaningful when placed in the context of network-level activity. As a result, combined AI/ML techniques are more accurate at detecting malicious activity and substantially reduce false positives compared to systems that use only a single source [27,29]. An integrated AI/ML architecture that combines network-level and endpoint-level telemetry is illustrated in Figure 2.1.

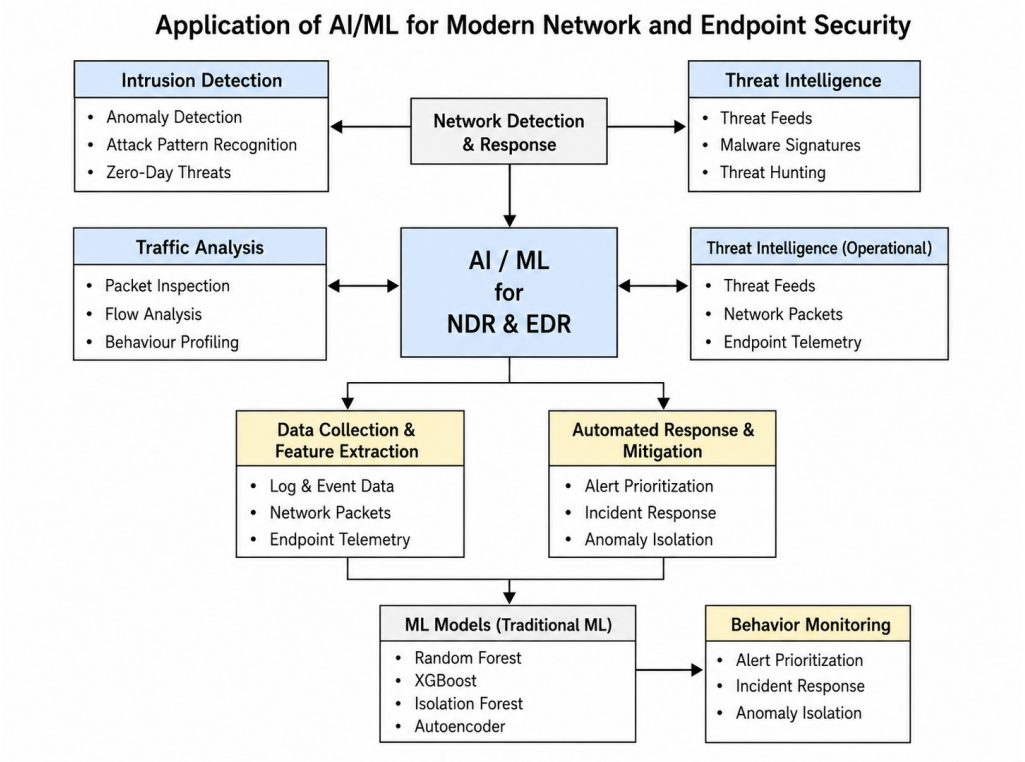

Figure 2.1: Integrated AI/ML architecture for modern NDR and EDR systems

2.7.4 Role of Large Language Models in Security Operations

Over the past few years, researchers have begun using LLMs to support security operations rather than as the primary means of intrusion detection [30]. In security workflows, LLMs are used to summarise alarms, turn low-level logs into natural-language explanations, reconstruct incident timelines, and support analysts with their investigations and reports. These features directly address problems such as alert fatigue and the mental strain experienced by security analysts [8,31].

LLMs help analysts quickly understand what happened and how critical it was by turning technical detection results and supporting information into a concise and easy-to-read explanation. This improves situational awareness of what is going on and can speed up triage and decision-making [32].

In most enterprise environments, network and endpoint data are collected through SIEM platforms. Traditionally, SIEMs use fixed, predefined rules to generate alerts, which are then sent to a SOAR platform for Tier-1 analysts to review. SIEMs rely heavily on alerts from multiple security tools. Another aggregated solution called XDR correlates data across endpoints, networks, and cloud environments. Then it provides a unified, end-to-end picture of the network. Recent evaluations by Oniagbi (2024) demonstrated that this approach can create a major bottleneck due to many fragmented alerts from cloud and network sources [33]. Consequently, many modern SOCs are considering adding LLM agents to the SOAR layer. LLM agents handle initial triage and provide context for SIEM alerts before human analysts review them.

2.8 Challenges of AI/ML and LLMs

Despite the extensive use of AI/ML and LLMs in NDR and EDR systems, many technical and operational issues persist, making them less useful in real-world security operations. These issues are particularly important for detecting lateral movement, as malicious activity often appears as regular and legitimate administrative tasks. Moreover, the evidence is distributed across network and endpoint telemetry [35]. These problems are outlined in this section to support the actual research objectives of this thesis. The main objective is to examine integrated NDR and EDR pipelines, evaluate their operational impact, and determine the role of LLMs as controlled post-detection reasoning components [36].

2.8.1 Gaps of AI/ML in NDR and EDR

The quality and reproducibility of the data. One of the major issues in machine learning-based intrusion detection is the lack of high-quality, clean, structured,

and reproducible datasets. Security telemetry is often noisy, incomplete, outdated, or incorrectly labelled. Public datasets may not accurately show how enterprise networks or endpoints behave. These issues make experimental results less reliable and challenging to compare. To address this, this thesis uses controlled datasets and clearly describes their features [25].

Dataset shift and limited generalisation. If ML models are trained on specific network conditions, they may not perform well in other settings. Model generalisation can degrade due to dataset shifts in user behaviour, infrastructure, and attacker tactics. This can reduce the ability to detect lateral movement and supports the use of behavioural features and cross-source correlation instead of environment-specific signatures. Another problem is class imbalance and the risk of false positives [29]. Intrusions are rare compared to typical network and endpoint activity. It can create a severe class imbalance. As a result, ML models could either miss attacks or produce too many false positives. This definitely makes analysts' work more challenging. This issue is directly relevant to one of the thesis goals, such as MTTD and MTTR, to assess their impact on operations.

ML-based alerts are difficult to understand. NDR and EDR systems typically generate alerts as anomaly scores or binary classifications; however, these explanations are often inadequate. Analysts must manually link and interpret these signals to understand the nature of an attack, which makes it challenging for them to analyse and respond to such attacks. This restriction creates a need for organised mechanisms for correlation and explanation, which are assessed in this thesis [37].

2.8.2 Challenges of Large Language Models in NDR/EDR Workflows

Hallucination and absence of evidentiary foundation. When provided with insufficient or ambiguous information, LLMs may produce explanations that appear plausible but lack supporting evidence. In security operations, these hallucinations can confuse analysts and slow response actions [38]. Sometimes, unstructured or unfiltered data, including outliers and garbage, can also cause hallucinations when making decisions. This informs the design decision in this thesis to limit LLM inputs to structured, evidence-based summaries [74].

Reasoning that may be repeated and checked In regulated situations, free-form reasoning over raw data makes it hard to duplicate and audit LLM outputs. Different prompts and model variants can reduce trust. This challenge directly affects the distinction between detection and interpretation as defined in this thesis [39].

Context selection and operational robustness The arrangement and presentation of detection outputs significantly affect the LLM's performance. Choosing the wrong context can lead to confusing or misleading explanations, which is why it is important to have clear incident construction and a standardised input schema [40]. Network and endpoint telemetry often contains sensitive information, such as usernames, hostnames, and internal IP addresses. Integrating raw logs into LLMs raises concerns about data privacy and compliance with applicable laws and regulations. This problem necessitates the use of sanitised, summarised incident representations rather than raw telemetry [41,42].

2.9 Scope and Delimitation

This thesis primarily focuses on identifying and analysing "lateral movement" and other internal attack behaviours that are common in the business environment. Specifically, the scope includes internal remote execution, unusual remote login patterns, unauthorised service creation, and credential abuse techniques that attackers often exploit after gaining access. These behaviours are chosen because they are key features of recent attack chains and are difficult to detect, as they employ real administrative tools and protocols [43]. The thesis focuses only on NDR and EDR telemetry, which are typically found in enterprise monitoring systems. This includes network flow data, protocol metadata, and endpoint event logs. The packet payload is intentionally omitted from inspection because it is typically encrypted and unavailable in the real world [44]. Relying on payload data would make the proposed approach less practical. Open-source, publicly available benchmark datasets are used for training, testing, and evaluation to avoid privacy issues in this experiment. This approach makes it easy to share results, repeat the process and compare with earlier work. These datasets provide a controlled environment for evaluating detection efficiency and integration methodologies. However, they may not fully reflect the operational challenges in a production environment [45]. In this thesis, the scope of work for LLMs is intentionally limited to reduce hallucination rates. Rather, LLMs are used solely as a layer for reasoning. It operates on structured incident summaries generated by NDR and EDR detectors using machine learning. In this thesis, the LLM does not change or replace the outputs of structured detection models. It does not process any raw packet payloads or unfiltered logs. This boundary enables the LLM to help analysts better understand investigations and accelerate them, while keeping the detection logic from becoming random, difficult to understand, or difficult to audit [30,33,87].

2.10 Ethics and Data Protection

Ethical standards and data protection regulations were central to the planning and execution of this research. Only publicly available and regulated experimental datasets were used, so any sensitive organisational or personal information was not handled. These datasets are standard in academic research, and no confidential data were involved in the experiment. These setups are designed to simulate attack behaviours without involving real users, production systems, or external networks. No real personal, organisational, or consumer data were collected or processed during the experiment, in accordance with the EU Directive (EU) 2022/2555 (NIS2 Directive) [46].

The use of LLMs was limited and grounded to ensure responsible and ethical use. LLMs received only structured event summaries that did not include personal details, such as usernames, hostnames, IP addresses, or file locations. No raw logs or packet data were provided to the LLM, reducing the likelihood of data leakage or accidental exposure [47].

2.11 Recent Studies and Advancements in Lateral Movement Detection and LLM-Assisted Security Operations

Recent studies on detecting lateral movement in enterprises have shifted away from signature-based and single-event anomaly detection toward graph-based, temporal, and multi-source modelling methods. King proposed EULER, which frames lateral movement detection as a scalable temporal link-prediction challenge within corporate authentication graphs, demonstrating improved detection efficacy while maintaining low false-positive rates across diverse settings [35]. Bowman also developed

an unsupervised graph-learning method to detect lateral movement in an organization's regular internal logs. These findings demonstrate that the relationships between hosts and accounts can identify stealthy and anomalous patterns without requiring labelled attack data [48]. Smiliotopoulos conducted a comprehensive investigation that classified lateral movement detection methodologies into signature-based, behaviour-based, and graph-based categories, concluding that relational and multi-stage strategies offer greater resilience against covert, multi-stage attacks [10]. Numerous studies indicate that standalone NDR and EDR deployments, as well as detection research efforts, exhibit structural limitations. Network-based systems are effective at identifying unusual east-west communication patterns, whereas endpoint systems provide detailed host telemetry, including process creation, logon activity, and service installation. However, operating these systems independently complicates data interpretation and results in fragmented alerts. This fragmentation impedes analysts' effectiveness and delays prioritisation. Turcotte argues in his study on Automated Alert Classification and Triage (AACT) that detection pipelines often fail not because of defective models, but because alerts lack the organised contextual enrichment necessary for human decision-making [32]. Additionally, research on SOC alert fatigue and human-AI collaboration shows that numerous minimally explained alerts reduce analyst effectiveness and increase MTTR. These results suggest that more than simply accurate detection models are needed to make operations go more efficiently. Therefore, specific tools are needed to integrate information from different sources.

Over the last few years, LLMs have emerged as tools that Security Operations Centres can use to support investigations, explain alerts, and summarise logs. Habibzadeh examined the application of LLMs in SOC workflows and found that they facilitate reporting and comprehension, but are highly sensitive to input structure and contextual factors [30]. Kramer studied LLM-assisted incident investigation

support and found that it improved report writing and provided advice on incident reasoning. They also found that hallucination and unwarranted belief in unfounded outputs posed significant risks [49]. These difficulties are consistent with findings from broader LLM security surveys: prompt injection, data leaks, and poor reasoning are serious problems in operational deployments.

A 2025 study by Singh found that analysts primarily utilise LLMs to interpret and contextualise fragmented logs, rather than for automated detection. The report also noted that even if LLMs are skilled at articulating complex events, analysts still struggle to trust them and to integrate them effectively with detection systems [50]. In 2025, Junaid conducted research assessing the linguistic capabilities of several LLMs to enable SOC automation [87]. A 2024 thesis paper by Oniagbi demonstrated that LLM agents can accelerate Tier-1 triage of security alarms in automated triage. Oniagbi's study used a zero-shot learning approach to classify Wazuh SIEM alerts and used them to provide LLMs with historical context [33]. This showed that these models can turn alert data into comprehensible summaries and help minimise MTTR. Retrieval-Augmented Generation (RAG) designs have been proposed to mitigate these issues.

Structured logs or knowledge bases are used to obtain the appropriate evidence before it is transmitted to the language model in these systems. Lewis formally introduced Retrieval-augmented generation (RAG) as a method for combining parametric language models with non-parametric retrieval to improve factual grounding. Recent implementations focused on SOC have shown that retrieval-based contextualisation significantly improves the reliability of security analytics procedures [51].

2.12 Limitations of Existing Research and Identified Gaps

Even with these advancements, significant research gaps remain. In particular, research on lateral movement detection has advanced through graph-based modelling, anomaly detection, and machine learning. These methods are effective at detecting unusual communication patterns, propagation behaviours, and authentication paths. However, most studies focus on detection performance, measured using metrics such as accuracy, F1-score, precision, and recall. While these metrics are important, they do not fully capture the real-world challenges that security analysts face.

Research on LLM-assisted security operations has also examined conversational reasoning support, automated report generation, and log summarisation. Many studies use loosely organised prompts and do not thoroughly assess grounding reliability or measure operational benefits, though these methods do improve readability and documentation efficiency. In addition, the effect of structured contextual correlation on the effectiveness of LLM-assisted reasoning in security processes remains underexplored.

Operational usability research differs from detection-focused research. While many studies have examined detection accuracy, fewer have examined how structured post-detection augmentation methods affect alert fragmentation, contextual organisation, interpretability, and response times.

Recent LLM integrations have demonstrated clear operational benefits. However, there is still room for architectural improvement. For example, LLM agents are often evaluated on existing security alert and event datasets. Although this shows that LLMs can summarise effectively, it does not address the main problem of alert fragmentation caused by raw NDR and EDR telemetry. Handling alerts individually, without accounting for correlations, simply shifts the issue of alert fatigue to the

LLM. RAG also has known issues. SOC agents often make retrieval errors, which can cause hallucinations or miss important context [33]. Singh (2025) also found that separating AI sense-making from primary detection helps with audits and maintains analyst trust [50]. This thesis proposes a three-layer architecture, rather than using an LLM to generate responses to individual preprocessed alerts via RAG. Before the LLM processes the fragmented telemetry, the raw ML detection results are combined using a deterministic Structural Correlation Mechanism. This method reduces retrieval-based hallucinations. It also ensures that the LLM serves as a transparent reasoning layer rather than a black-box detection tool by using rigorous Grounded Prompting based on these correlated concepts rather than RAG [51]. This method removes retrieval-related errors and reduces the risk of hallucination by limiting the LLM input to structured, correlated evidence.

In summary, this thesis addresses the research gap by developing and evaluating a post-detection approach that combines grounded LLM-assisted reasoning with structured contextual correlation. The main focus is to analyse how adding context affects operational efficiency, such as reducing the number of alerts, making investigations clearer, and improving MTTD and MTTR, rather than creating new detection algorithms. Table 2.1 provides a comparative overview of existing lateral movement detection methods and modern LLM-assisted SOC approaches, highlighting their respective advantages and limitations.

Technique	**Principle**	**Advantages**	**Limitations**
Signature-Based Detection	Matches predefined attack patterns (e.g., known SMB/RDP abuse signatures).	Low computational cost; high precision for known threats.	Ineffective against novel or stealthy lateral movement; requires frequent updates.
Statistical / Anomaly-Based Detection	Detects deviations from baseline authentication or network behaviour.	Detects unknown attacks; adaptable to the environment.	High false positives; limited interpretability.
Graph-Based Lateral Movement Detection	Models enterprise entities (hosts, users) as graphs and detects anomalous paths or link predictions.	Captures multi-stage attacks; scalable to large networks.	Computational complexity requires structured telemetry.
NDR-Based Detection	Analyses east–west traffic patterns, protocol logs, and flow anomalies.	Network-wide visibility; detects abnormal propagation patterns.	Limited host context; encryption reduces visibility.
EDR-Based Detection	Monitors endpoint telemetry (processes, logons, service creation).	Detailed host-level insight; confirms execution artifacts.	Lacks cross-network propagation context; alert fragmentation.
Integrated NDR-EDR Correlation	Combines network and endpoint telemetry for cross-layer analysis.	Reduces blind spots; improves detection confidence.	Increased system complexity; data fusion challenges.
Automated Alert Triage Systems	Classifies and prioritises alerts using machine learning filtering pipelines.	Reduces analyst workload; improves prioritisation efficiency.	Often lacks explainability; dependent on feature engineering.
LLM-Based Log Summarisation	Uses large language models to translate logs into human-readable explanations.	Improves interpretability; reduces cognitive load.	Sensitive to prompt quality; hallucination risk.
Hybrid Detection + LLM Contextual Analysis (Proposed Direction)	Structured detection (NDR/EDR/-graph) followed by context-grounded LLM explanation and prioritisation.	Enhances interpretability; reduces false positives; improves MTTR.	Requires secure architecture; careful context engineering is needed.

Table 2.1: Comparative Overview of Lateral Movement Detection and LLM-Assisted SOC Approaches

3 Methodology

This chapter presents a methodology that combines deterministic alert correlation with the functionality of grounded LLMs to support incident triage and enhance interpretability. It begins by outlining the system workflow and explaining how different alerts are grouped from separate entities. This step is crucial to turning raw telemetry into coherent security incidents. The chapter then discusses design decisions, such as reducing the risk of hallucinations, while fulfilling the practical needs of SOC analysts. Finally, this section highlights the role of LLMs in generating evidence-based summaries.

3.1 System Architecture Overview

The methodology builds on the concepts discussed of Chapter 2 and provides a modular design. This architecture decouples contextual reasoning from statistical detection. The detection flow is shown below:

Raw Telemetry → Detection → Correlation → Grounded LLM Reasoning

Raw Telemetry: NDR and EDR sensors continuously collect network traffic and endpoint logs.

Detection: Machine learning models such as XGBoost and Autoencoders evaluate telemetry data to identify anomalies. They trigger specific security alerts when

suspicious behaviour is detected.

Correlation: A deterministic engine addresses alert fragmentation by grouping similar alerts into structured, organised incident bundles based on shared entities.

Grounded LLM Reasoning: The incident bundle is structured and then sent to an LLM with grounding and strict rules. The model then automatically converts numerical indicators into a readable incident narrative.

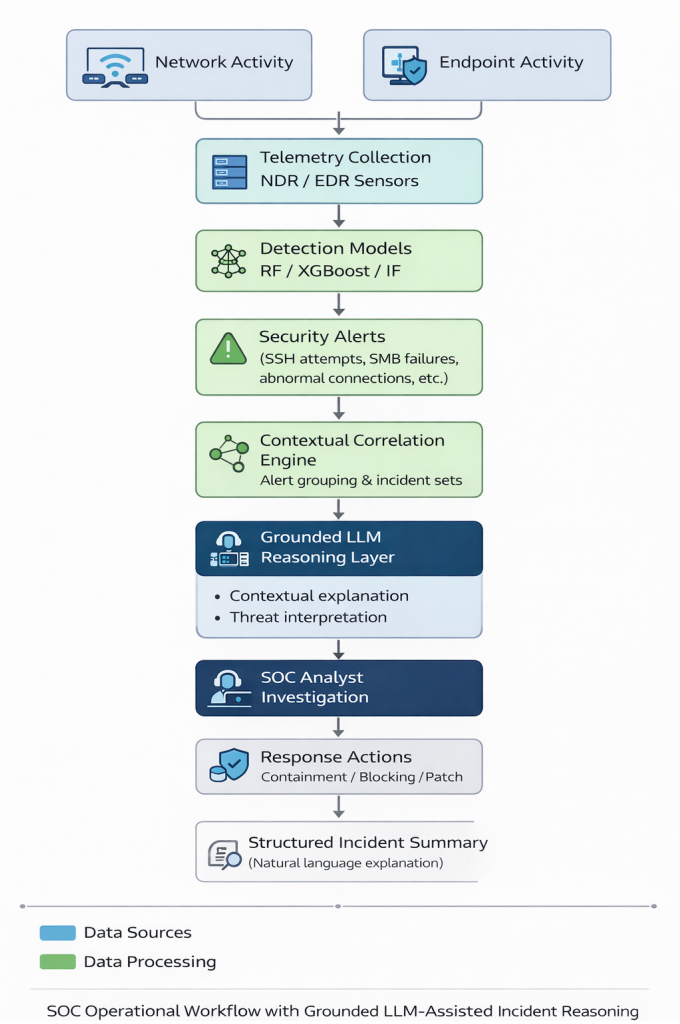

Figure 3.1: Proposed SOC operational workflow

Figure 3.1 shows the proposed SOC framework. Network and endpoint operations generate telemetry that monitoring systems such as NDR and EDR collect. The telemetry streams are monitored using machine-learning detection models, including Random Forest, XGBoost, Isolation Forest, and Autoencoder-based anomaly detectors [52].

If the detection models find suspicious activity, such as repeated SSH login attempts, failed SMB connections, unusual lateral movement, or unexpected traffic patterns, they trigger alarms. These notifications are then grouped using a contextual correlation method. This method groups similar events into organised incident bundles based on their similarity and temporal proximity [53,56].

The grounded LLM reasoning layer takes structured event reports and generates natural-language explanations that demonstrate the alerts' essential role in behaviour. This reasoning layer assists SOC analysts by simplifying and consolidating alerts while preserving statistical detection methods [54,55].

Each layer makes a controlled change. The detection stage identifies statistical differences. The correlation stage groups alerts into logical incident structures. The LLM stage makes it easier to understand without altering the detection decisions. This strong separation ensures that the results can be reproduced and that there is no contamination between the statistical and generative components [56].

3.1.1 Operational Context of Security Alert Generation

SOCs continuously receive security alerts from streams of telemetry data generated by network infrastructure and endpoint devices. NDR systems monitor network traffic, including connection flows, authentication attempts, and protocol exchanges, across an enterprise network. EDR systems enhance monitoring by collecting host-level activity data, including process creation, attempted user logins, file access, and privilege escalation [57].

Figure 3.2: Sample SIEM alert dashboard representing typical SOC monitoring interfaces used for incident detection and investigation

In Fig. 3.2, the SIEM dashboard illustrates security alerts and incident monitoring. The interface displays multiple alert categories, including SSH brute-force attempts, failed authentication events, malware detections, and suspicious data transfer activities. Such alerts represent the type of telemetry analysed by the proposed detection and contextual reasoning framework.

In most cases, NDR and EDR platforms capture telemetry as low-level log entries rather than as alerts that can be acted on immediately. This type of telemetry includes connection logs, authentication records, session metadata, and statistics on host communication. Each record includes the source and destination addresses, the service type, the connection duration, the authentication results, and the protocol's response codes [58]. Detection engines take these raw data streams and convert them into structured feature representations that statistical or machine-learning al-

gorithms can use. For example, if a host repeatedly attempts to log in via SSH within a short period, this could suggest an attacker is attempting to gain access via brute-force methods. Similarly, if someone repeatedly attempts to access SMB services from multiple internal hosts, this could indicate lateral movement following a compromise. Suspicious Remote Desktop Protocol (RDP) connections between internal hosts could also indicate that someone is attempting to access the network without permission or to spread malware within it. Most of the time, these behavioural indicators are based on numerical values, including the number of connections, failed authentications, host interactions, and protocol-specific error rates [59].

Detection models examine observed behaviour and assess its deviation from typical patterns after extracting relevant features from the telemetry stream [2,63]. Machine learning classifiers and algorithms for anomaly detection produce predictions or anomaly scores that indicate the likelihood of malicious activity. When these outputs exceed the specified detection thresholds, the monitoring system sends a security notification indicating the observed event [60].

This methodology focuses on testing the pipeline in a controlled environment using static datasets in this experiment, but the architecture is designed for use in a live enterprise SOC for future deployment. In production, the framework works as an automated microservice layer between a distributed telemetry data broker and a SOAR platform. To manage large volumes of internal (East-West) network traffic, including core routers, switches, perimeter firewalls, application delivery controllers, and endpoint agents, data must be ingested, buffered, and routed efficiently. In a production environment, this is implemented with a high-throughput event streaming platform such as Apache Kafka. Network sensors and endpoint agents send their logs as raw data to a specific partitioned Kafka [61].

In a future production deployment, the ML detection layer works as a scalable, real-time Kafka consumer. It subscribes to telemetry topics and accumulates the required numerical features as data comes in. Then it applies trained models, such as XGBoost or an Autoencoder, directly to the stream. Once it detects an anomaly, the ML microservice sends a structured "raw alert" payload to a dedicated output topic. The correlation engine can operate alongside a SOAR platform [53,56]. The pipeline uses a deterministic Decision-Level Fusion layer to reduce false-positive alerts. This provides the Tier-1 analyst with a clear, evidence-based summary in the alert or incident management dashboard. As a result, the pipeline becomes a scalable, real-time stream-processing system rather than only a static batch process. In a typical SOC operation flow, when an anomaly is detected, the ML microservice sends alerts and starts an Automated Enrichment Pipeline [64]. If EDR identifies a new or unknown file hash or NDR identifies a suspicious external IP address, the system can simultaneously send REST API requests to cloud reputation services such as Cisco Talos and VirusTotal [41]. If the indicator is not recognised, which could indicate a zero-day threat, the file is sent to a cloud sandbox, such as Palo Alto WildFire, for dynamic analysis, or to an on-premises sandboxing solution, such as the Cisco Threat Grid Appliance.

Subsequently, the SOAR platform collects these enriched artefacts by combining the original ML alert with reputation scores from the cloud and sandbox reports. It groups these related events into a single JSON schema. This provides the Enterprise LLM with a 'Grounded' data packet that includes both internal telemetry and global threat intelligence [81]. In this way, the summary shown to the analyst is grounded in internal telemetry and external threat intelligence, supported by up-to-date global context from the cloud. The visual representation of this end-to-end SOC workflow in terms of lateral movement is illustrated below in Figure 3.3:

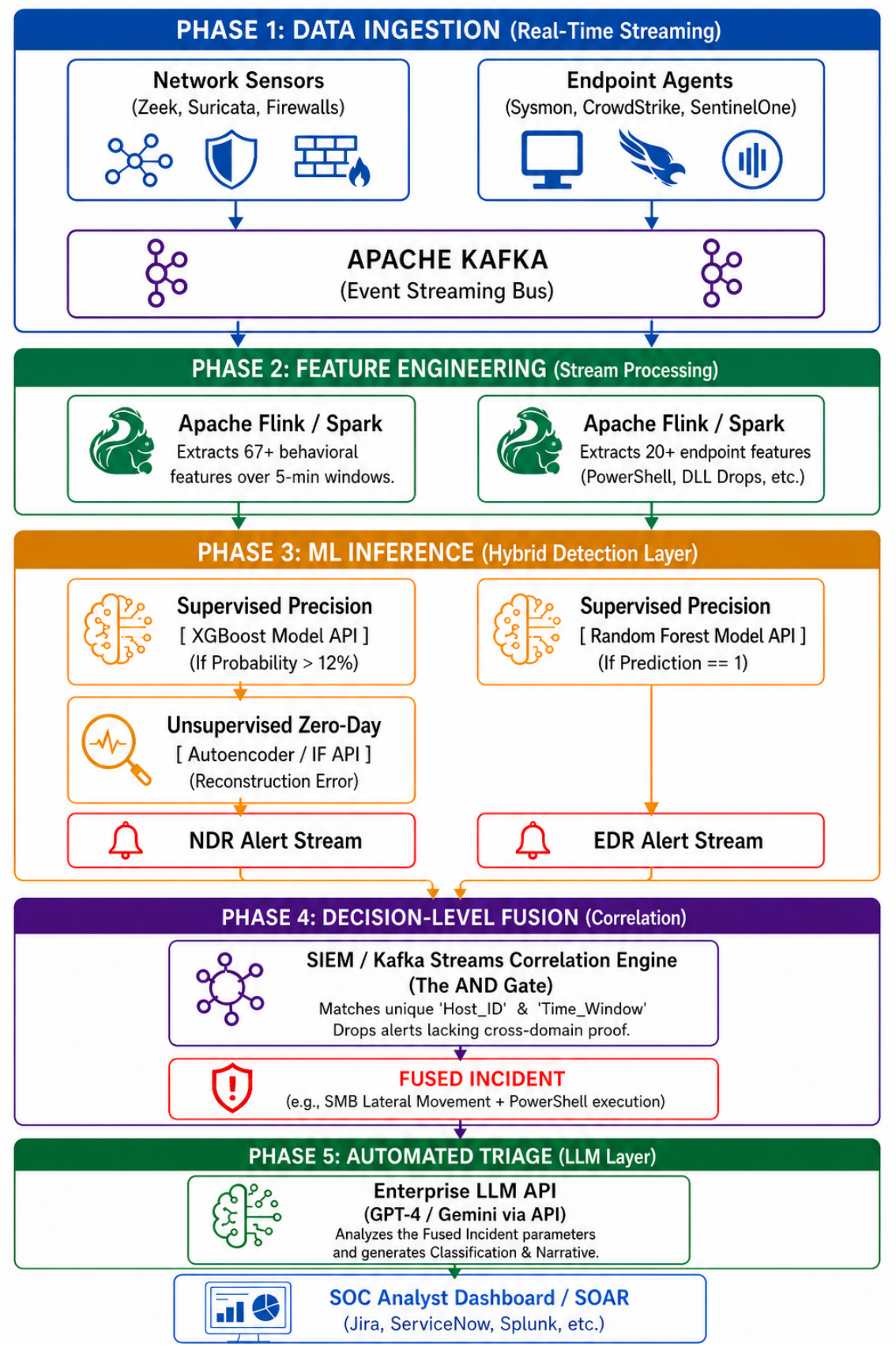

Figure 3.3: Enterprise NDR/EDR pipeline with hybrid ML inference and automated triage.

3.2 Comparison with existing literature

The framework in this chapter addresses the limitations discussed in Sections 2.9 and 2.10. Recent studies on lateral movement detection mainly focus on machine learning and behavioural anomaly detection. These methods can help identify unusual authentication paths and internal communication patterns. Techniques like temporal graph modelling, relational feature extraction, and unsupervised anomaly detection are effective in the case of detecting malicious activities. However, as mentioned in the literature review, most studies evaluate detection systems mainly based on statistical metrics such as precision, recall, and F1-score. These measurements are useful for evaluating classification performance. However, the objective of this thesis is to face the real-world challenges faced by security analysts in SOC [66]. NDR and EDR systems generate a large volume of alerts that analysts are often trying to manage. If there is such an incident that contains multiple alerts, such as multiple records with no context, then the analysts must assemble an incident summary from the scattered signals [67].

Recent studies on LLM-assisted security operations have investigated how LLMs can help with log summarisation, answering analysts' queries and automated reporting. These approaches make documentation and support for analysts relatively better. However, many studies rely on loosely organised prompts and do not thoroughly evaluate the impact of structured inputs on the reliability of reasoning [30].

The methodological design of this thesis focuses on mitigating these constraints. It implements a layered analytical approach that distinguishes statistical detection from contextual reasoning. Machine learning models use structured telemetry features to make detection outputs. Then, a deterministic contextual correlation method groups related alerts into incident bundles based on shared entities and temporal proximity. Lastly, a grounded LLM reasoning layer uses these structured incident descriptions to generate contextual explanations and suggestions [55].

The proposed methodology assesses the potential of contextual enrichment to enhance the operational usability of NDR and EDR systems while preserving their statistical detection performance by integrating detection modelling, structured alert correlation, and grounded generative reasoning within a controlled pipeline. [85]

3.3 Detection Modelling Framework

The first methodological layer of this research develops a reliable detection backbone that acts as the foundation for all subsequent contextual enrichment and reasoning phases. Previous research on intrusion detection has primarily focused on categorisation accuracy, rather than the operational effects of detection outputs. Therefore, before structured contextual correlation and grounded LLM-assisted reasoning can be implemented, it is important to develop and evaluate a reliable detection layer. This ensures that statistical behaviour, optimisation method, and performance characteristics are carefully controlled and understood.

This thesis does not propose a new method for detecting any new anomalies. Methodologically, the comparison of paradigms seeks to ensure a stable and effective baseline classifier. By rigorously evaluating detection performance at the outset, the study distinguishes the impact of contextual enrichment from pure classification variance.

Let the dataset be represented as:

$$D = \{(x_i, y_i)\}_{i=1}^{N} \tag{3.1}$$

where $x_i \in \mathbb{R}^d$ represents structured feature vectors derived from telemetry and $y_i \in \{0, 1\}$ denotes binary labels which refer to normal versus attacks.

The detection objective is to learn:

$$f : \mathbb{R}^d \rightarrow \{0, 1\} \tag{3.2}$$

that distinguishes statistically benign behaviour from abnormal or harmful variations.

3.3.1 Behavioural Interpretation of Features

The dataset includes structured, numeric, and categorical attributes. However, these features are not treated as abstract statistical variables. Rather, they are understood as reflecting behavioural patterns in network and host interactions. Connection-level attributes such as duration, protocol type, service type, and connection flags capture sessions. Traffic-volume indicators, such as source and destination bytes, reflect communication intensity and asymmetry. Statistical measures such as connection count, service count, and host-based repetition help identify patterns of persistence, concentration, and scanning behaviour.

In cybersecurity, lateral movement and internal reconnaissance typically present as repeated login attempts, unusual service checks, more frequent host-to-host communication, or unusual error patterns. Even if the dataset does not explicitly label "lateral movement," its statistical patterns encode proxy manifestations of such behaviours. Therefore, detection modelling focuses on identifying unusual changes or deviations in structured telemetry that match patterns of internal pivoting or suspicious propagation, as described in this thesis.

3.3.2 Supervised Ensemble Models: Random Forest and XGBoost

Two supervised ensemble methods are used in the experiment: Random Forest and Extreme Gradient Boosting (XGBoost). The reason for choosing these models is

that they are based on distinct ensemble philosophies, yet both are good starting points for structured intrusion detection tasks.

Random Forest

Random Forest is an ensemble of decision trees that operates similarly to a bagging system [69]. Each tree is trained on a bootstrapped sample of the dataset, and at each split, a random subset of features is considered. This randomness reduces variance and improves generalisation, especially in network telemetry, where feature spaces are often very high-dimensional.

The Gini index is often used to figure out the impurity of a decision tree node:

$$Gini = 1 - \sum_{c} p_c^2 \tag{3.3}$$

Final classification is determined by majority voting:

$$\hat{y} = \text{mode}(T_1(x), T_2(x), \ldots, T_B(x)) \tag{3.4}$$

where p_c is the proportion of class c, $T_b(x)$ is the prediction of b-th tree and $\hat{y}$ is the final class with majority voting. For the purposes of this thesis, Random Forest is particularly effective at capturing nonlinear feature interactions. For instance, a large number of connections by itself might not indicate anything abnormal, but when combined with high error rates and unusual service access patterns, it could suggest suspicious internal activity. Tree-based partitioning naturally illustrates how these interactions operate. Cross-validation on the training data only is used to select the best hyperparameter values, such as the number of trees, the maximum depth, and the minimum samples per split. Because of class imbalance and the sensitivity of operations to both false positives and false negatives, the optimisation objective prioritises the F1-score.

XGBoost

XGBoost usually represents a boosting-based ensemble model. XGBoost does not grow trees on its own. It starts with a base prediction and then finds the residual errors, which are the exact dissimilarities between what it predicted and what really happened. Instead of teaching new trees how to predict what the final goal is, it teaches each new tree how to guess and rectify the errors that all the trees that came before it made. XGBoost continues to improve at what it does by rectifying errors hundreds of times and utilising strong mathematical regularisation to prevent overfitting. This iterative error correction approach is effective at identifying difficult, non-linear patterns in tabular data [62]. For instance, it can traverse large amounts of network traffic to discover rare malicious behaviour. Instead, it develops trees sequentially, each time making a regularised objective function smaller:

$$L = \sum_{i=1}^{N} l(y_i, \hat{y}_i) + \sum_{k} \Omega(f_k) \tag{3.5}$$

where $l(\cdot)$ denotes logistic loss and $\Omega(\cdot)$ penalises model complexity. Each new tree fixes the fault left by the previous group. XGBoost is highly effective at identifying small changes in structured traffic patterns, including low-frequency anomalies that constitute only a small fraction of the dataset but nevertheless exhibit anomalous behaviour.

Grid search is used to tune hyperparameters, such as the learning rate, tree depth, subsample ratio, and regularisation coefficients, on the training dataset. Optimisation focuses on Precision-Recall AUC since it is stable under class imbalance [58].

3.3.3 Unsupervised Paradigms: Isolation Forest and Autoencoder

To avoid over-reliance on labelled attack patterns, two unsupervised paradigms are incorporated: Isolation Forest and Autoencoder.

Isolation Forest

The Isolation Forest, an unsupervised anomaly-detection technique, assumes that malicious events are "rare and different." Instead of characterising normal network traffic, it actively distinguishes each packet from the rest of the dataset by randomly selecting features and partitioning the data. Because attacks and anomalous behaviour are statistically rare and far from typical traffic, they are quickly segregated and end up at the very top of the trees with limited path lengths. The system computes an anomaly score based on how easily a data point can be isolated [70]. This makes it very fast and effective at finding previously unseen attacks without requiring labelled training data. The anomaly score is derived from expected path length:

$$s(x) = 2^{-\frac{E(h(x))}{c(n)}} \tag{3.6}$$

where s is the anomaly score, $E(h(x))$ denotes the expected path length, and the normalising factor is $c(\psi)$. Anomalous samples typically require fewer splits for separation, resulting in shorter path lengths and elevated anomaly scores. This thesis examines Isolation Forest as a method for simulating the ability to detect unknown vulnerabilities and compares its performance with supervised classification.

3.3.4 Autoencoder-Based Reconstruction

An Autoencoder is a type of deep learning neural network that learns the underlying mathematical structure of a regular system without any explicit attack labels [57]. It works in two steps: first, it compresses raw network telemetry into a small, low-dimensional bottleneck (the Encoder). Then, it tries to reconstruct the original data from that compressed state (the Decoder). The model needs to be trained on normal, benign traffic to become very good at reconstructing benign packets. However, when it is suddenly subjected to unusual network traffic, the network does not know how to properly compress or reconstruct it [71]. This leads to a large "reconstruction error," and if the error exceeds a particular threshold, the model immediately flags the packet as an anomaly. The Autoencoder is trained exclusively on normal traffic to learn a compressed latent representation of benign behaviour. For input x_i, reconstruction $\hat{x}_i$ is generated. Reconstruction error:

$$E_i = \|x_i - \hat{x}_i\|_2^2 \tag{3.7}$$

serves as an anomaly indicator. A high reconstruction error means that the learned normal structure is not being followed. This representation-learning method is well-suited to tree-based methods because it models nonlinear feature dependencies in the latent space.

3.3.5 Controlled Experimental Protocol

To prevent data leakage:

- Pre-processing parameters are fitted exclusively on training data.
- Hyperparameter tuning is conducted via cross-validation within the training set.
- Final evaluation is performed on an unseen test set.

The detection stage produces structured alerts:

$$A = \{a_1, a_2, ..., a_m\} \tag{3.8}$$

Each alert a_i contains a predicted label, a confidence score, a timestamp, and entity identifiers. These alerts represent record-level detection events and form the raw material for contextual processing.

To improve the reliability of the machine learning evaluation, cross-validation was applied to the detection layer. The fixed correlated event bundles generated from the held-out test data were used to evaluate the LLM's reasoning performance. The cross-validation findings were employed as a reliability check for the detection layer, and a fixed 80/20 train-test split was used for the final correlation and LLM-assisted reasoning experiments. This separation allowed evaluation of model stability without affecting the downstream incident correlation and contextual reasoning procedures.

At this stage, detection performance is evaluated using precision, recall, F1-score, and balanced accuracy. However, these metrics assess only statistical classification quality; they do not address alert organisation or interpretability. This detection modelling framework is illustrated in Fig. 3 .4 below :

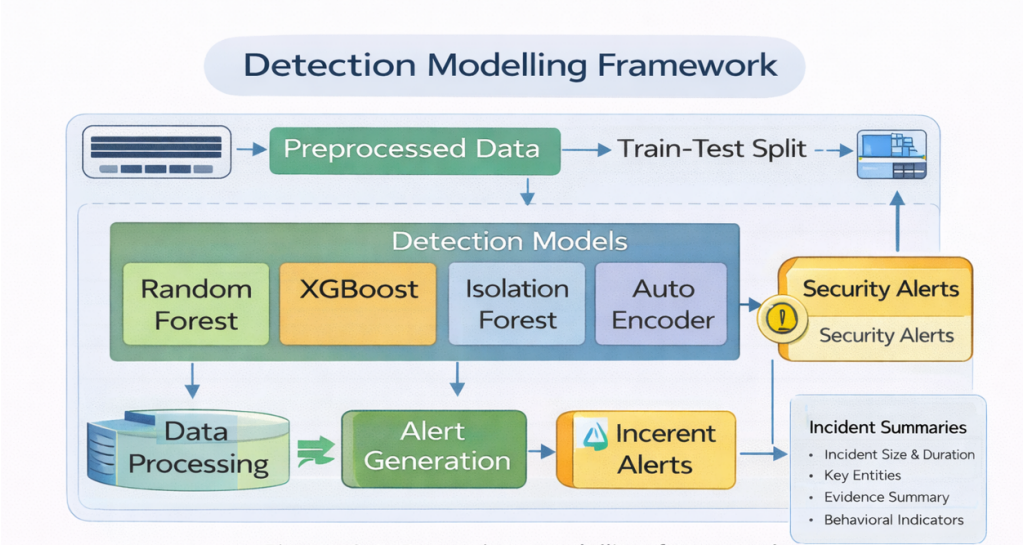

Figure 3.4: Detection Modelling Framework

3.4 Structured Contextual Correlation Mechanism

Detection models operate on individual records, whereas security analysts consider incidents. This mismatch causes alert fragmentation, in which many record-level detections typically point to the same malicious action. Without structural aggregation, analysts must manually piece together the broader picture of the incident [72].

To address this structural gap, this thesis presents a deterministic contextual correlation mechanism that lies between detection and LLM-assisted reasoning [73].

Let the alert set be:

$$A = \{a_1, a_2, ..., a_m\} \tag{3.9}$$

Each alert a_i contains timestamp t_i, entity identifiers e_i such as hosts, users, or IP address and confidence score $\hat{s}_i$.

If in a live SOC environment, two alerts are considered related, they must have at least one entity identifier in common, and the following condition must be met:

$$|t_i - t_j| \leq \Delta t \tag{3.10}$$

Alerts form nodes in a graph. Similarity relations form edges. Connected components represent incident bundles:

$$I = \{I_1, I_2, ..., I_k\} \tag{3.11}$$

Each incident bundle aggregates:

- **Number of alerts:** This shows how many times the incident was detected and is a measure of how intense the behaviour was.
- **Dominant behavioural indicators:** These are the indications in the bundle that are the most statistically significant or common, and they sum up the incident's behavioural signature.
- **Average detection confidence:** This measure adds up the model confidence scores for all alarms to give a numerical indication of how suspicious the timestamps and entity identifiers are.

In this thesis, the Alert-to-Incident Ratio (AIR) technique was used to measure, from an operational perspective. It can also measure the average number of repeated or common detections compressed into a single ticket. Based on this mathematical representation, the ratio of correctly correlated events to related events was shown, thereby reducing alert volume. The Alert-to-Incident Ratio is an adaptation of Maosa's defined ratio that measures the operational efficiency of the correlation engine [80]. In a coherent narrative, the greater the ratio, the more efficient the reduction in SOC analyst burden. This Alert-to-Incident Ratio concept is explored in detail in the Result Analysis and Evaluation chapter (5.2.1). In this thesis, two independent public datasets (CIC-IDS-2018 for NDR and Empire APT3 Sysmon for

EDR) were used. Since these do not have synchronised timelines, the correlation engine applied a synthetic and deterministic mapping in this experiment. Alerts were matched across domains by simulating target ports and using semantic mappings to meet the target ports, behavioural scenarios and timing requirements. This enabled the study to measure the correlation engine's performance in reducing alerts. Moreover, the performance of the LLM is mathematically evaluated based on true-positive detections. This method allowed to recreate a real SOC-type environment without altering the actual attack patterns. It enabled the testing of an automated triage layer realistically.

Although the machine learning models found real anomalies in these datasets, public repositories do not naturally generate the heavy alert traffic in real enterprise cyberattacks. Figure 3.5 presents the overall evaluation framework for this study, comprising both the alert correlation pipeline and the LLM contextualisation phase.

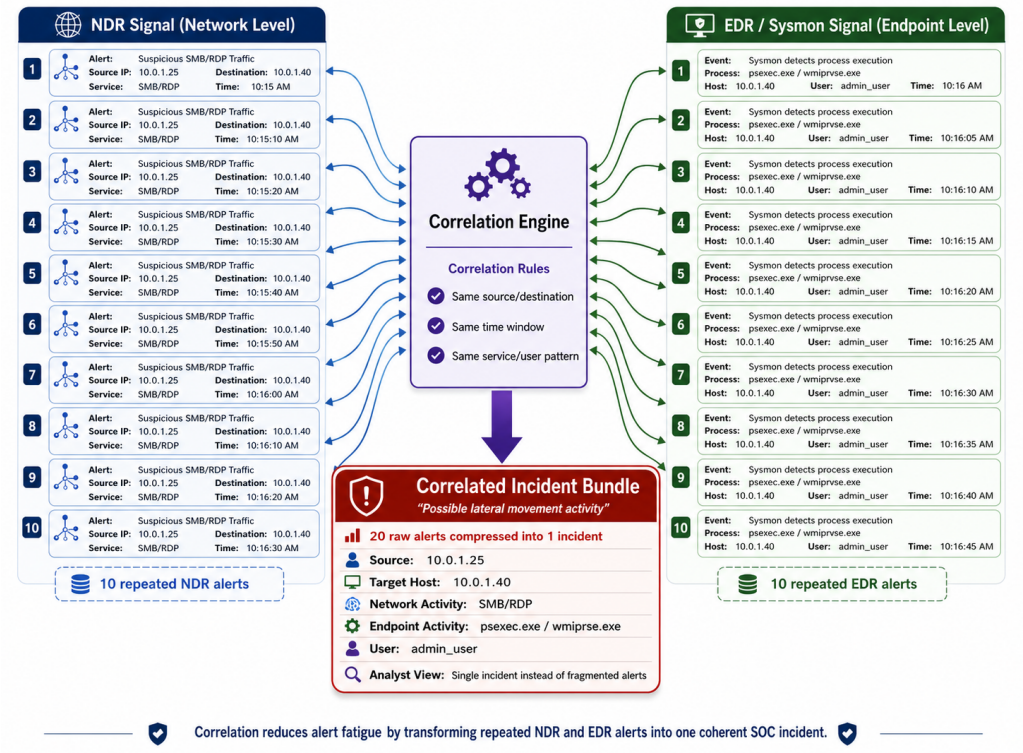

Figure 3.5: Structured Contextual Correlation Mechanism

3.5 Grounded LLM-Assisted Contextual Analysis

The detection and correlation layers help to enhance alert reliability and make alerts easier to find. However, they cannot fully address the challenge that security analysts face in interpreting these alerts. Even when alerts are grouped, analysts still need to review the combined data and determine its implications for operations. This transition from structured data to clear explanations is a key reason why the SOC team often respond slowly [32,88].

The final step in this thesis introduces a grounded LLM as a reasoning assistant after detection. The LLM does not detect anomalies or affect how events are categorised. Instead, it adds an interpretive layer that turns organised event reports into explanations that analysts can easily understand [73]. This difference is highly significant for the research design, ensuring novelty. The main objective of this thesis is not to replace traditional machine learning detectors with generative models. Instead, it explores how adding context and grounded reasoning can make operations more useful and structured while maintaining results that are repeatable and clear.

3.5.1 Conceptual Role of the LLM in the Architecture

During the detection step, statistical models can determine whether a record is inconsistent with learned normal patterns. In the correlation phase, structurally related alerts are linked and combined into incidents. At this point, the algorithm has produced a structured representation of potentially malicious activities, but are oriented to the machine.

However, structured data such as a high reject rate, an elevated connection count, service-mismatch patterns, and host-based anomaly ratios remain numerical abstractions. Analysts need to determine what these combinations imply for an attacker's behaviour. The LLM enters the picture precisely at this point of interpretation. It

consumes structured summaries and generates:

- A concise narrative explanation of the seen behaviour.
- Connection between behavioural indicators and possible attack hypotheses.
- Explicit statements of uncertainty.
- Recommended steps for further investigation

Thus, the LLM serves more as a reasoning interface than a classifier. It demonstrates how an LLM can turn plain alert data into a clear, easy-to-read triage report for security teams by analysing and summarising the information in the figure 3.6

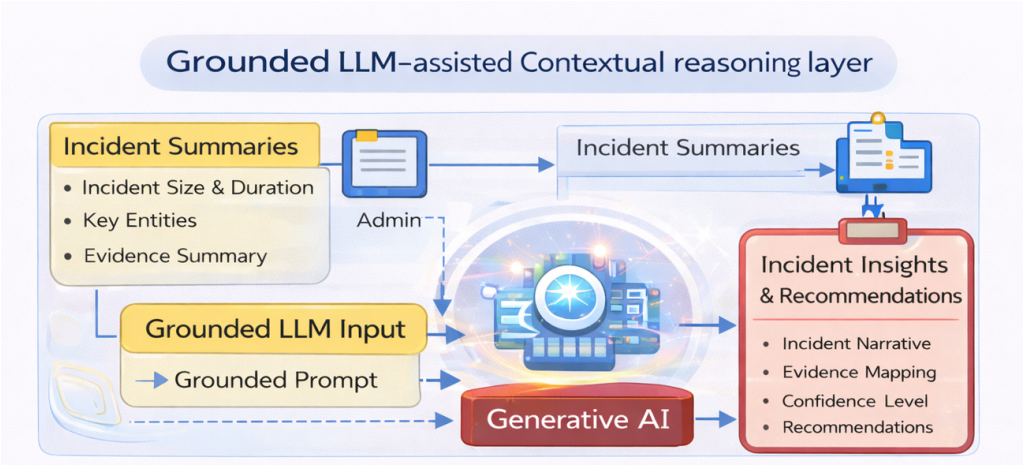

Figure 3.6: Grounded LLM-Assisted Contextual Reasoning Layer

3.5.2 Structured Input Schema and Grounding Constraints

A major concern when using LLMs in cybersecurity is the risk of hallucination. Without adequate regulation and loaded raw, unfiltered alerts, these generative models may hallucinate attacker identities, historical details and unsupported verdicts [38].

In this thesis, the LLM strictly constrained the grounding rules to avoid such issues. The model does not see raw logs or packet data. Instead, it is provided with

a structured event schema. Then the correlation layer appears, which includes only confirmed numerical and categorical fields from the detection outputs.

Let an incident bundle be represented as:

$$S_r = \{n_{\text{alerts}},\ time_{\text{span}},\ key_{\text{entities}},\ mean_{\text{score}},\ dominant_{\text{features}}\} \tag{3.12}$$

The LLM input is generated only from S_r. No external retrieval, browsing, or raw telemetry is provided.

The prompt template enforces the following rules:

- All explanations must use the summary fields.
- No external assumptions are allowed.
- If there is no essential information, the model has to confirm the uncertainty.
- The output must clearly distinguish between evidence and assumption.

This design aligns with the rules of grounded reasoning and ensures that generated outputs can always be traced back to structured evidence.

3.5.3 Interpretability and Cognitive Load Reduction

Even after identifying correlations, analysts still need to use their judgment to construct a causal explanation from the combined data. A structured summary could show, for example:

- 42 related alerts
- Elevated `srv_count`
- High `error_rate`
- Multiple destination services contacted

- Consistent anomaly score above threshold

Experienced analysts know these statistics are important. However, they still need to interpret them to form an operational hypothesis. The LLM handles this translation automatically. It transforms numerical summaries into meaningful reasoning while preserving uncertainty. This reduces alert fatigue, cognitive overhead, context-switching effort and manual documentation time. Thus, the detection accuracy remains unchanged. However, the clarity of interpretation improves.

This thesis makes a fundamental methodological choice to keep statistical detection and generative reasoning clearly separated. It does not change the anomaly scores or go back to the detection thresholds. This separation ensures that improved detection accuracy is not confused with improvements in factors, for example, the ease of understanding results or the clarity of triage.

This also helps make results reproducible. If the detection outputs stay the same, the LLM's results depend only on organised summaries and fixed prompts.

3.5.4 Operational Impact and Proxies for MTTD/MTTR

The grounded LLM reasoning layer does not change the statistical detection boundary. Instead, it directly affects the SOC operations. Measures such as precision and recall quantify the statistical reliability of alerts. On the other hand, operational performance is typically measured by the time required to detect and respond, using metrics such as MTTD and MTTR [63].

This experiment uses offline historical telemetry rather than a live SOC deployment, making it impractical to capture actual chronological timing. This thesis proposes measurable proxy metrics for assessing simulated operational impact:

The pipeline provides a mathematical reduction of the separate investigations an analyst must perform by measuring how many fragmented record-level alerts are combined into incident bundles (e.g., compressing 50 raw alerts into 5 structured

incidents). A higher compression ratio can potentially reduce the MTTR [72].

The grounded LLM automates the translation of numerical aggregation into semantic reasoning. The system eliminates the need to manually compose queries during an investigation, quickly combining dominant behavioural signals, clear uncertainty, and suggested triage measures. This reduction in cognitive burden and the effort required to transition contexts is a proxy for reducing triage time, such as MTTD [8,76].

3.5.5 Limitations of LLM Integration

Even with grounding constraints, LLM-based reasoning is still relying on chance. Depending on the generation parameters, the outputs may differ slightly each time. To reduce variability:

- Temperature settings are fixed.
- The format of the prompt is the same every time.
- The input schema stays consistent.

Additionally, the LLM is not a reliable method for identifying the attacker. It suggests ideas but does not provide final answers. The SOC team remains responsible for the final decision. This approach maintains ethical and operational boundaries while leveraging generative AI to add helpful context.

4 Dataset Preparation and Experiment

This chapter explains the experimental setup. CIC-IDS-2018 and Empire APT3 datasets were used to simulate enterprise-style attack scenarios. It begins with an overview of the experimental environment, followed by the reason for the selection of datasets. This creates a fundamental process for combining network and endpoint anomalies. Next, the chapter covers data engineering tasks, including synthetic correlation processes such as timestamp alignment, log mapping, and telemetry organisation. Finally, it demonstrates how these datasets help create reliable LLM test cases to measure system performance.

4.1 Experimental Environment and Setup

To evaluate the proposed multi-stage detection and contextual reasoning framework, a controlled lab environment was established to simulate a SOC workflow. In this setup, supervised and unsupervised ML models handled statistical detection, while LLMs provided post-detection contextual enrichment.

Initially, a cloud-based platform was developed to enable rapid testing and prototyping of the machine learning pipeline. Then, the system was transferred to a secure, locally hosted LLM on the university network to test the limits of privacy-preserving deployment, similar to those in enterprise SOC setups. The same struc-

tured and correlated alerts were fed to publicly accessible cloud LLMs. This hybrid approach demonstrates that the proposed architecture can operate across both cloud and local environments and can be applied in real-world cybersecurity scenarios. Table 4.1 below presents the lab environment and tools used in this experiment.

Component	**Cloud Prototyping**	**Secure Local Deployment**
Compute Environment	Jupyter Notebook (Anaconda)	Jupyter Notebook (Anaconda)
Data Processing	Python (Pandas, NumPy)	Python (Pandas, NumPy)
Machine Learning	Scikit-learn, XGBoost, RF, Autoencoder	XGBoost, Random Forest, Autoencoder
LLM Engine	OpenAI API (gpt-4o, Gemini)	Open WebUI
Security Model	Public HTTPS API	No API key/Token

Table 4.1: Experimental environment configurations for cloud-based prototyping and secure local deployment

4.2 Dataset Selection and Preparation

In practice, anomaly detection performance depends primarily on how the input data are represented while maintaining quality. This simulation approximates real SOC conditions, where different data types are analysed together from multiple sources, especially SIEM systems. For this scenario, two independent public data sources, CIC-IDS-2018 and Sysmon(Empire APT3), were used to provide network-level and endpoint-level telemetry. Using both datasets aligns with how modern SOCs operate, along with NDR and EDR running concurrently to provide a comprehensive view of the infrastructure.

4.2.1 Network Dataset: CIC-IDS-2018

For the network-based detection segment, the dataset utilised was CIC-IDS-2018. The Canadian Institute for Cybersecurity developed this dataset [78]. It contains genuine, realistic network traffic with regular benign behaviour, along with a variety of attack vectors, including lateral-movement-related behaviours, brute-force, denial-of-service (DoS), infiltration, and command-and-control activities.

To ensure statistical significance and prevent model overfitting, a robust sample of 100,000 network flow records (rows) was employed, while keeping computational cost within the constraints of the local inference process. The data were stratified and split into 80% for training the behavioural baselines (80,000 samples) and 20% for testing and alert production (20,000 samples).

In this experiment, the processed dataset contained around 67 features. Feeding all raw variables into a machine learning model directly poses a high risk of target leakage. A strict feature selection process was applied to prevent models from memorising the identities of specific attackers and to instead learn the mechanics of the attack. The machine learning training did not include explicit contextual identifiers such as Source and Destination IPs, MAC addresses, Flow IDs, or timestamps. Instead, detection models were trained only on strictly behavioural and statistical aspects, such as:

- Flow duration and Inter-Arrival Times (IAT)
- Forward and backward packet length variance (mean, max, min, standard deviation)
- TCP/UDP flag counts and ratios
- Flow byte rates and packet sizes

By constraining the feature space to these attributes, the detection models must identify mathematical signatures of malicious activity, such as persistent scanning

behaviour, rapid traffic growth, or erratic data flows, rather than relying on static indicators of compromise [78]. This enables the system to generalise to previously unseen attack patterns from unknown source addresses. Contextual features such as destination ports were retained in the larger dataset and reintroduced in the downstream correlation and fusion layer to map network alerts to their corresponding EDR telemetry.

4.2.2 Endpoint Dataset: Empire APT3 Sysmon Telemetry

Perfectly synchronised, high-quality datasets containing both Network (NDR) and host-level (EDR) telemetry for the same attack vectors are rare in cybersecurity academic research. To evaluate the pipeline's ability for cross-domain reasoning and decision-level fusion, the Empire APT3 dataset was used in this experiment [47].

To train the EDR detection models, the Empire APT3 datasets were selected, as they are real Microsoft Sysmon event logs mapped to MITRE ATT&CK created in the context of a realistic Advanced Persistent Threat (APT) emulation [79]. These logs were handled at the stream engineering layer, resulting in approximately 7,600 rows and 20+ endpoint behavioural features. The machine learning layer examines real-world forensic artefacts in endpoint telemetry to validate network-level anomalies. Key monitored behaviours simulated in this dataset include:

- **Event ID 1 (Process Creation):** Helps to detect unusual parent-child process relationships, such as *svchost.exe* launching *mstsc.exe* during an RDP anomaly, or *bash.exe* spawning *sh.exe* during an SSH brute-force attempt.
- **Event ID 3 (Network Connection):** Identifies when specific binaries, such as *ssh.exe*, initiate outbound connections to external IP addresses.
- **Command-Line Auditing:** Tracks malicious administrative actions, such as PowerShell.exe bypassing execution policies to run SMB scripts.

4.2.3 Data Preprocessing

Both datasets were subjected to preprocessing steps to ensure compatibility with machine learning models:

- **Data Cleaning:** Removal of missing and infinite values from raw telemetry.
- **Feature Selection:** Selection of relevant features capturing behavioural anomalies such as connection frequency and error rates.
- **Normalisation:** Scaling of numerical features to maintain consistent value ranges.
- **Encoding:** Conversion of categorical attributes into numerical form.

4.2.4 Structural Interpretability and Feature Attribution

The original behavioural characteristics of the dataset, such as “Packet Length”, “IAT Std”, and “Flow Duration”, make them highly applicable to advanced feature-attribution techniques and LLM-driven natural-language explanations.

This structural interpretability is crucial for downstream correlation and generative AI in the pipeline. For instance, the architecture enables an LLM to interpret the combined mathematical results of an event in terms of intelligible domain-specific patterns, such as high IAT variance with repetition, TCP packet flags, or specialised port scanning. Using these interpretable metrics, the LLM constructs an incident narrative that explains why a specific fused flow was labelled as an attack or anomaly. This explanation layer is much more useful and operationally relevant when applied to structured datasets with inherently interpretable behavioural features [39,81].

4.2.5 Decision-Level Fusion: Correlating NDR and EDR Context

At this stage of the experiment, a deterministic synthetic correlation process was used to combine NDR and EDR alerts independently. The main goal was to reconcile the abstract network statistics and the semantic context of the endpoints, as the public datasets do not contain synchronised timestamps or shared host identifiers. In the proposed architecture, this phase acts as a logical "AND Gate" to correlate telemetry. The Decision-Level Fusion stage of the pipeline combines these different telemetry sources. The architecture applies synthetic mapping rules and relies on real, ATT&CK-aligned datasets, such as Empire APT3 and high-fidelity network captures, to enforce a deterministic logical AND-gate. Cross-domain correlation is validated using a synthetic mapping rather than shared host identifiers. This behavioural and temporal validation ensures that only high-fidelity, evidence-backed incidents are forwarded to the downstream LLM triage layer. It helps reduce false-positive alerts in high-sensitivity machine-learning models.

The fusion engine does not randomly add context when the ML inference layer labels a raw network flow as abnormal. Instead, it applies a deterministic synthetic mapping to identify Empire APT3 Sysmon alerts, which are based on predefined attack scenarios and targeted service ports within the overlapping *Time_ Window*. The cross-domain correlations assessed in this experiment are:

- **Port 3389 (RDP) & Event ID 1:** Network-level RDP scanning correlated with endpoint logs showing *mstsc.exe* launched by an unauthorised parent process.
- **Port 445 / 139 (SMB) & Command-Line Activity:** Network-level anomalous SMB transfers validated by endpoint execution of *powershell.exe -ExecutionPolicy Bypass*.

- **Port 22 (SSH) & Event ID 3:** Network-level SSH brute-force patterns confirmed by endpoint logs of *ssh.exe* initiating rapid, repeated outbound connections.
- **Ports 80 / 443 (Web) & Event ID 1:** Web protocol scanning behaviour correlated with the execution of scanning binaries (e.g., *nmap.exe*) on the host system.

This deterministic fusion aims to ensure that the automated triage layer using an LLM produces a highly organised, evidence-based schema. The pipeline reduces false positives by requiring that network anomalies be supported by comparable endpoint data from the Empire APT3 emulation, thus simulating a key aspect of the SOC environment.

4.3 Machine Learning Pipeline Implementation

The detection layer is the first layer of the proposed design. Its task is to detect statistically unusual or malicious behaviour in structured telemetry data. This layer operates independently, separate from the next two stages of correlation and LLM reasoning. This enables testing detection performance in isolation.

Both supervised and unsupervised models were used to get different detection capabilities. Supervised models, such as XGBoost, learn from labelled data on known attack patterns. Unsupervised models, on the other hand, seek deviations from normal behaviour, analogous to identifying zero-day attacks [69,70]. The pipeline uses a hybrid approach. Unsupervised models review NDR telemetry to detect unknown network anomalies without relying on signatures. At the same time, supervised models check EDR telemetry to reliably spot known malicious payload executions.

Model	**Hyperparameter**	**Value**	**Purpose**
XGBoost	`max_depth`	2	Aggressive regularisation to prevent memorisation
XGBoost	`decision_threshold`	0.12	Shifted to introduce realistic false-positive variance
Random Forest	`max_depth`	3	Restricts tree complexity for broad generalisation
Random Forest	`min_samples_leaf`	10	Forces generalised behavioural boundaries
Isolation Forest	`n_estimators`	50	Balances ensemble robustness and processing speed
Autoencoder	Reconstruction error threshold	95th percentile	Flags high-reconstruction-error samples as anomalies

Table 4.2: ML model hyperparameter configuration for generalised detection

The hyperparameter settings are summarised in Table 4.2 and were selected to mitigate the common problem of model overfitting. This is a major challenge when working with highly structured datasets such as CIC-IDS-2018 [77]. Rather than allowing the supervised models to evolve into complex decision trees that memorise the dataset, strict regularisation parameters were applied. To encourage the models to learn general, underlying attack behaviour rather than small, dataset-specific noise, the hyperparameters (max depth) for XGBoost and Random Forest were set to 2 and 3, respectively, and a high minimum leaf size was set.

In addition, a deliberate shift of the threshold in the supervised models (i.e., lowering the XGBoost decision boundary to 0.12). In a real-world SOC, a detection

baseline must balance security with a realistic level of volatility. By lowering this threshold, a controlled rate of false positives was deliberately introduced, reducing precision below 100%.

For unsupervised anomaly detection, a neural network autoencoder with a bottleneck architecture was constructed to compress and reconstruct the latent representations of strictly benign data. The anomalies were then categorised using the Mean Squared Error (MSE) of the reconstruction loss relative to dynamically optimised percentiles, in conjunction with the Isolation Forest. This multi-layer architecture ensures that only highly reliable, generalisable behavioural alarms are sent to the downstream correlation layer.

An 80/20 train-test split was used in the experimental pipeline for final alert generation and downstream correlation. Moreover, the detection models were tested for stability using five-fold stratified cross-validation. The purpose of cross-validation was not to generate LLM inputs but to confirm that the performance of supervised detection was not dependent on a single random dataset split. The final test split was retained to create the alert-level output, which was then fed into the deterministic correlation engine and subsequently into the grounded LLM reasoning layer.

4.4 Alert Correlation Engine Setup

The detection models produce alert-level outputs, but they remain fragmented and do not immediately relate to higher-level security problems. This fragmentation makes it much harder for analysts to interpret incidents efficiently because many alerts are typically about the same threat. In the experiment, alert volumes or the number of alerts generated were automatically generated using a mathematical scaling method applied to the time window integer 'x' for validated true-positive detections. NDR and EDR alert counts were calculated with the formulas $x \pmod{35}+12$ and $x \pmod{85}+15$, respectively. This approach helped to simulate high and var-

ied alert volumes that often occur between network and endpoint sensors during a simulated cyberattack scenario. For instance, when a true-positive attack occurs during a validated attack time window, the pipeline uses predefined formulas to calculate the alert volume. The EDR alert volume is $47 \pmod{85} + 15$, which gives 62 alerts. At the same time, the NDR alert volume is $23 \pmod{35} + 12$ resulting in 35 alerts. This method automatically generates unique and realistic alert combinations for each incident bundle. It can reduce the need for manual effort. The fusion process initially identifies time windows in which both Network Detection and Response (NDR) and Endpoint Detection and Response (EDR) alerts co-occur, utilising an AND-gate as described in Appendix A.4-A.7 . Subsequently, the pipeline determines or assigns the relevant network service port, which serves as a reference for mapping distinct scenarios. For example, Port 445/SMB is associated with potential lateral movement involving suspicious PowerShell activity. Port 3389 with remote desktop activity, Port 22 with SSH-based remote logins, Port 53 with suspicious DNS queries, and Ports 80/443 with suspicious outbound connections. A deterministic correlation engine was used to combine relevant alerts into organised incident bundles to get around this problem. The public datasets did not include shared entity identifiers such as IP addresses or hostnames. That's why, in this proof-of-concept fusion approach, the alerts were grouped by overlapping time windows, matching scenario validation, and the targeted network service port.

This change reduced duplicate alerts and provided a more coherent picture of security occurrences, which will be used as input by the next LLM reasoning stage.

$$AIR = \frac{|A|}{|I|} \tag{4.1}$$

where $|A|$ is the total number of alerts and $|I|$ is the number of events that are connected. This metric measures the reduction in the number of fragmented alerts. An automated, multi-factor fusion engine was used in this framework to address the

limitations of isolated detection systems and reduce alert fragmentation. Rather than using timestamps solely, the system applies a clear 'AND Gate' logic in three steps.

First, separate machine learning models analyse network traffic (NDR) and host telemetry (EDR). They generate binary alerts based on learned behaviour thresholds.

Next, the fusion engine receives these alert streams and applies a strict three-part correlation. Alerts must meet all three conditions to pass through the fusion gate:

- **Temporal Alignment:** The NDR and EDR alerts must occur within the same overlapping time window.
- **Semantic Behavioural Mapping:** The detected network anomaly must logically correspond to the observed endpoint artefact.
- **Scenario Validation:** Both alerts need to correspond to the same type of attack vector. For example, if there is a network-level SMB anomaly on Port 445, it should be supported by a host-level PowerShell execution policy bypass. Since the experiment uses two different datasets, the pipeline uses the targeted network port as the primary reference to match each network anomaly with its related endpoint scenario.

  The following figure 4.1 illustrates how this process works. Finally, if the telemetry meets all the conditions, the engine combines the signals into one reliable Incident Bundle. If an anomaly appears in only one area, such as network noise without corresponding endpoint activity, the fusion gate rejects it. This helps reduce false positives and sends only confirmed incidents to the next LLM triage step, enabling the LLM to perform better and avoid the risk of hallucinations.

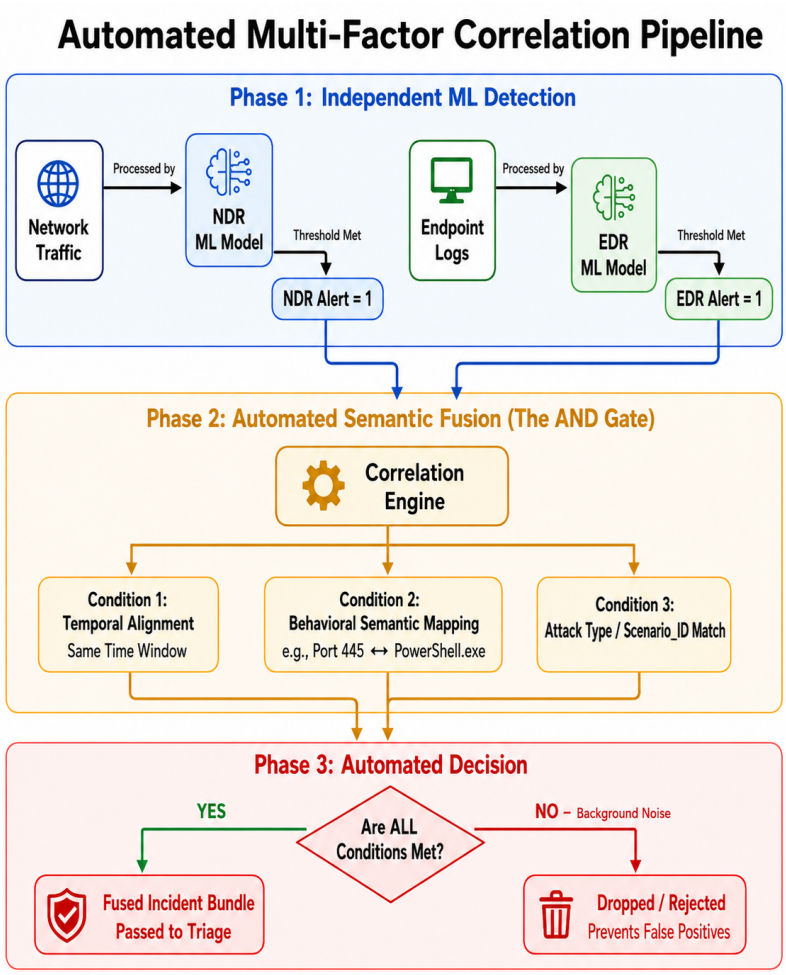

Figure 4.1: Automated Multi-Factor Correlation Pipeline for NDR and EDR telemetry fusion.

4.5 Large Language Model (LLM) Integration

The final step in the experimental pipeline adds a Large Language Model as a reasoning tool after detection. The LLM does not classify data as detection models do; instead, it gives context-based interpretations of related incident data [81]. Two deployment strategies were used to evaluate both performance and feasibility in practice.

4.5.1 Cloud-Based Deployment

The first version used the OpenAI API to access a lightweight LLM. The correlation engine generated structured JSON summaries that served as input, along with a limited prompt template that required evidence-based reasoning. This approach made it easy to quickly check the reasoning pipeline while keeping the output format the same.

4.5.2 Privacy-Preserving Local Deployment

The system was deployed to a locally hosted LLM infrastructure within the university network to support the privacy of enterprise data. The implementation used an Open WebUI interface linked to an Ollama-based inference engine that hosted models such as LLaMA 3.2 and Gemma 3. This deployment demonstrates that the proposed architecture can operate in air-gapped or restricted contexts without sending critical security telemetry to services outside the network.

4.5.3 Model Selection and Specifications

For the final evaluation, the architecture's reasoning abilities and its performance in practice were assessed using four different Large Language Models. These models were chosen to show a range of deployment modes (cloud vs local) and parameter sizes (heavyweight vs lightweight distillation) [33].

- **GPT-4o (OpenAI) and Gemini-2.5 (Flash):** This was chosen as the commercial cloud baseline. It can be accessed via a highly optimised, lightweight REST API. It has strong natural language understanding. For SOC narratives, OpenAI (GPT-4o) and Gemini are considered the best platforms for maintaining the highest linguistic quality. However, storing sensitive information in the cloud is not ideal for an organisation.

- **Gemma 3 (27B):** Chosen as the main local candidate with a lot of parameters. Google developed this open-source model, which has 27 billion parameters and can perform deep reasoning comparable to commercial cloud APIs. It was decided to assess whether a heavyweight model could handle complex, fully on-premises fused telemetry.

- **Llama 3.2 (3B):** Chosen as the local option that is very lightweight. This open-weight model has 3 billion parameters was [50]. It was added to the project to assess whether a much smaller model could still produce useful, grounded forensic narratives without placing excessive strain on the university's infrastructure.

The experiment shows the distinction between the models' architectures (API vs Local) and parameter scales (27B vs 3B) in terms of operational delay and reasoning quality in a SOC environment, comparing these four models.

5 Result Analysis and Evaluation

This chapter provides a detailed evaluation of the proposed multi-layered framework. This framework builds on previous chapters and sets the foundation for a detailed evaluation. The focus is on four major aspects. First, the statistical detection performance (how accurately the framework identifies significant patterns using quantitative methods). Next, the structural efficiency (how well the framework's components interact), interpretability (how easily the output of the framework can be explained or understood by human users), and finally operational feasibility (how practical the framework is for real-world implementation). The analysis is structured around five main areas: i) Detecting anomalies by ML models, ii) Correlation of multiple alerts, iii) Performance of LLM in reasoning tasks, iv) Efficiency of the framework and v) Understandability of the analysis for the users. Each subsequent section builds upon the outcomes discussed previously to present a cohesive evaluation.

5.1 Machine Learning Detection Layer Performance

The initial layer of the proposed architecture evaluates the effectiveness of various machine learning models for detecting anomalous or malicious network behaviour. Traditional evaluation metrics, including accuracy, precision, recall, and F1-score, are employed. Unlike traditional IDS systems that rely on static signatures, this method evaluates the behavioural characteristics of network telemetry, particularly for lateral traffic movement. To reduce the risk of bias due to small sample sizes

and to achieve statistical significance, models are tested on a 100,000-row sample extracted from the CIC-IDS-2018, which focuses on lateral movement, and from the Empire APT3 Sysmon as endpoint datasets. Cross-validation results indicate whether the detection models perform consistently across multiple data splits. However, these results should be interpreted as a reliability check for the detection layer only. The downstream correlation and LLM-assisted reasoning stages were evaluated on the fixed held-out test set to ensure the generated alert and incident bundles were consistent.

Since accuracy sometimes behaves like a misleading indicator in cybersecurity, where it can be boosted artificially by the large volume of benign traffic, the models need to be examined using stricter metrics to indicate their practical value [83]:

Accuracy (Overall Correctness): The models achieved a high accuracy for the supervised model: >99% and the unsupervised model: >85% on average. However, this metric alone is often insufficient due to class imbalance [82]. An immature model may achieve high accuracy by simply predicting "benign" at all times.

Detection Rate (Recall): The percentage of real attacks detected as expected. The unsupervised models also performed well (Isolation Forest: 0.6787, Autoencoder: 0.9960). It shows that it detected a high proportion of malicious packets without prior signature knowledge.

Precision (Alert Fidelity): Precision measures the proportion of true attacks among all generated alerts. The threshold-shifting technique set XGBoost at a realistic operating precision of 0.9853. However, unsupervised models had lower precision (0.4152 and 0.6679), reflecting the trade-off inherent to novelty detection.

F1-Score (Harmonic Mean): A balance between Precision and Recall. The supervised models achieved excellent F1-scores (0.99), indicating good performance in the case of false-positive alarms [84]. The unsupervised models performed well (F1-scores around 0.6 on average), since the subsequent correlation layer will filter

out the remaining packet-level noise.

The results are summarised in Table 5.1, where supervised and unsupervised models are directly compared. Supervised models achieved relatively better metrics and reliability across all key measures. In contrast, the unsupervised models (Isolation Forest and Autoencoder) present the typical profile for novelty detection. These models learned benign baselines and flagged most abnormalities, thereby capturing unknown patterns. Their accuracy of packet-level precision was lower, between 55% and 59%. This happened because they sometimes misidentified unusual benign traffic. The correlation and fusion layer tried to solve this by grouping noisy raw alerts and assembling them into structured incidents. After that, these events went to the LLM for contextual analysis and reasoning. It enabled both methods to work together within the framework.

Model Type	**Accuracy**	**Precision**	**Recall**	**F1-Score**
XGBoost	0.9978	0.9853	1.00	0.9926
Random Forest	0.9990	0.9973	0.9957	0.9965
Isolation Forest	0.8070	0.4152	0.6787	0.5152
Autoencoder	0.9245	0.6679	0.9960	0.7796

Table 5.1: Optimised Performance Metrics achieved on samples from NDR and EDR of the regularised Dataset

To illustrate the statistical data presented in Figure 5.1, a comparative bar chart of the evaluation metrics across the four tested models is presented.

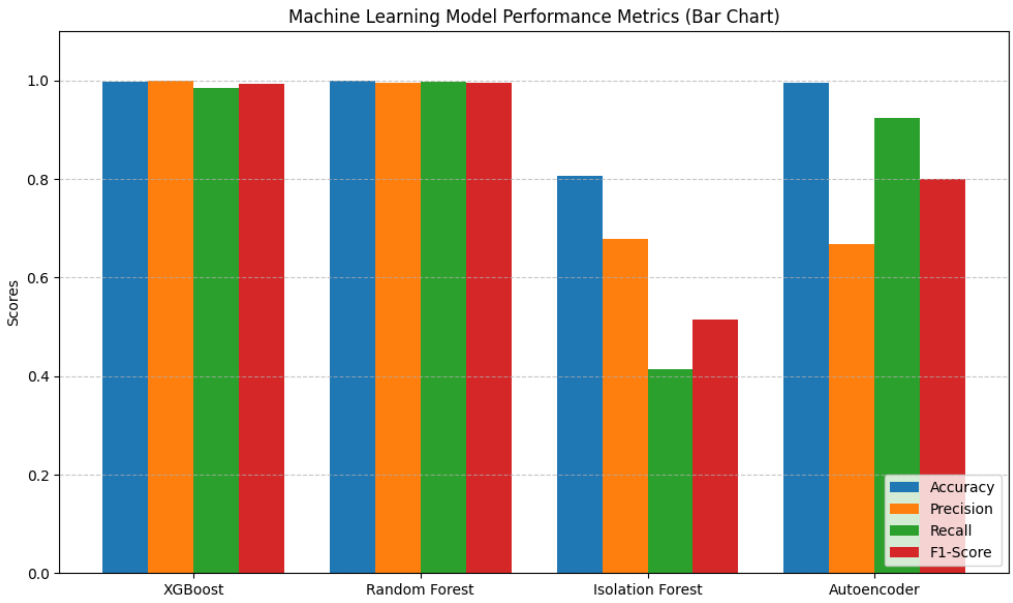

Figure 5.1: Machine Learning Model Performance Metrics Comparison

The grouped bar chart visibly highlights the operational trade-offs of the modelling paradigms. The supervised models (XGBoost and Random Forest) showed a near-perfect balance across Accuracy, Precision, Recall, and F1-Score. In contrast, the unsupervised models (Isolation Forest and Autoencoder) exhibit the predicted variance. The figure above clearly visualises the steep decline in Precision for these unsupervised models. It accurately reflects their role as high-sensitivity novelty detectors, which mark all deviations from the established benign baseline before downstream correlation.

5.1.1 Patterns of Attack Behaviour and Model Reliability

The performance of the supervised tree-based models demonstrates the effectiveness of the XGBoost and ensemble learning. Characterisation of telemetry data from a high-dimensional network was required. To ensure models do not overfit to fixed environmental identifiers, a strict feature-engineering approach was used to encourage them to focus on generalisable behavioural signatures. During training, certain

network identifiers, such as source and destination IP addresses, were removed to enable the models to focus on the mathematical mechanics of the traffic flows.

This regularisation ensures that the achieved XGBoost F1-score of 0.9926 is a reliable indicator that detection is not merely overfitting. Recall of 1.00 is especially important in cybersecurity, as it lowers the chance of false negatives that could expose networks to compromise. The F1-score, a balanced harmonic mean, mathematically confirms the robustness of this detection layer.

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{5.1}$$

The detection layer performed robustly on the 100,000-row CIC-IDS-2018 sample set for NDR. It can detect traffic patterns relevant to lateral movement, such as an attacker's attempt to move through a network from one device to another, using packet-level patterns. Thus, it remains effective even if attackers use non-standard ports or change network configurations during attacks.

5.1.2 Unsupervised Resilience and Anomaly Baseline Modelling

One important takeaway at this level was the robustness of unsupervised approaches. Autoencoder (accuracy: 92.45%) and Isolation Forest (accuracy: 80.7%) are among the key safety measures for detecting unknown anomalies. The Autoencoder was trained on a benign baseline containing selected rows from NDR traffic and learned to recognise regular traffic topologies. During evaluation, brute-force anomalies yielded substantially higher reconstruction errors, such as MSE, and enabled the system to detect malicious behaviour without requiring prior knowledge of attack labels. This capability ensures that the framework is effective against novel lateral movement strategies not present in the labelled training sets. The unsupervised

layer is the first line of defence against zero-day vulnerabilities when supervised labels are unavailable. Standard performance metrics were calculated by comparing the models' blind anomaly predictions with the concealed ground truth labels in the test dataset.

5.1.3 Feature Importance and Behavioural Decision Logic

A detailed feature-importance analysis was performed to evaluate the machine-learning layer's decision-making process. This analysis identifies the network telemetry attributes that most effectively discriminate between attacks and benign traffic. The results are consistent with the standards set for the CIC-IDS2018 dataset and suggest that the models focus on a mixture of temporal, volumetric, and statistical variables. The following characteristic categories were found as the main drivers of detection:

- **Flow Duration and Temporal Dynamics:** Apart from duration, the model also analyses *Flow IAT Mean* and *Fwd IAT Max*. Automated brute-force scripts have a mechanical, repetitive pace distinct from the unpredictable, "bursty" behaviour of human-initiated administrative sessions.

- **Packet Length Statistics (Max, Min, Mean):** Features like *Fwd Packet Length Max* and *Bwd Packet Length Std* were very important. Malicious lateral movement frequently employs uniform packet sizes (e.g., short login attempts or large specialised transfers) that yield a statistical distribution distinct from the diverse packet lengths of regular online or file-sharing traffic.

- **Bidirectional Volumetric Ratios:** The model computes the ratio of *Total Fwd Packets* and *Total Backwards Packets*. In a brute-force attack, there is an obvious asymmetry, such as outbound login attempts are high in number,

and the “Access Denied” responses are very small and uniform, a pattern that the XGBoost model can identify with high precision.

- **Flag Counts and Header Inspection:** The presence of certain TCP flags (e.g. SYN, ACK, or PSH) was a significant signal. Phases of discovery of lateral movement often leave a trail of certain flag sequences that are mathematically distinguishable from a completed, successful three-way handshake in a benign session.

By focusing on these multi-dimensional behavioural attributes rather than static network identifiers, such as IP addresses, the detection layer becomes highly operationally adaptable. This allows the NDR and EDR systems to identify threats based on the functional *action* performed, rather than the *identity* of the source, thereby fulfilling the "verify everything" criterion of modern Zero-Trust networks.

5.2 Structured Correlation and Alert Reduction

One of the main challenges in modern threat detection is alert fragmentation. Most detection models analyse single records, but real attacks involve thousands of correlated events. This discrepancy leads to alert fatigue, in which analysts become overwhelmed by repeated log entries [72,80].

5.2.1 Quantitative Volume Compression and AIR Metrics

The second layer of the system uses a structural correlation engine to solve this problem. Its efficiency is measured by the Alert-to-Incident Ratio (AIR), defined in equation 4.1, which quantifies the amount of information compressed. During testing, the machine-learning layer generated approximately 575 raw combined NDR and EDR alerts. Machine learning models can detect real anomalies in datasets, but public repositories do not generate the heavy alert traffic seen during real enterprise

cyberattacks. To properly test the reduction of alerts (Alert-to-Incident Ratio) by the correlation engine, the number of true-positive detections was increased via volumetric extrapolation. It also validates whether the LLM can handle a large amount of data. This approach allowed the proposed pipeline to simulate the alert volume of a real SOC environment without altering the actual attack patterns.

The 575 alerts (NDR+EDR) were grouped into 6 incident bundles using a correlation-based logic. It combines temporal closeness and target-service similarity. This means that alert compression efficiency keeps analysts focused on the entire incident narratives with recommendations rather than sorting through each log.

$$AIR = \frac{\text{Total Raw Alerts (575)}}{\text{Total Fused Incidents (6)}} \approx 95 \tag{5.2}$$

This level of compression is crucial to keep the operations manageable while reducing the workload. Otherwise, analysts would have to review all 575 alerts manually, which would take hours even at 30 seconds per alert. This is just a theoretical assumption. However, by grouping and correlating all these alerts, the system makes the analyst's job easier. It reduces their cognitive load, ensuring that important threats are not missed amid high-volume telemetry logs.

The final layer uses LLMs to generate human-readable narratives from the fused incident data. The study assessed both cloud-based models (GPT-4o, Gemini) and locally hosted models (Llama 3, Gemma 3) to assess the trade-off between reasoning depth and data privacy. Table 5.2 provides a sample summary of the correlated NDR-EDR incident bundles, detailing the number of alerts, the main targeted ports, and the classified threat behaviours observed in each event. The detailed report generated by LLM is available in Appendix B.

Incident ID	Alert Count	Dominant Port	Classified Threat Behaviour
1001	99	445 (SMB)	Unusual Internal Data Discovery
1002	82	80 (HTTP)	Internal Web Protocol Reconnaissance
1003	96	443 (HTTPS)	Secure Web Protocol Reconnaissance
1004	118	22 (SSH)	High-Intensity Credential Brute-Force
1005	90	3389 (RDP)	Remote Access Target Scanning
1006	90	53 (DNS)	Anomalous DNS Query / Tunnelling

Table 5.2: Summary of Fused Incidents and Alert Counts

5.2.2 Contextual Enrichment via NDR-EDR Data Fusion

The main strength of this framework is that it combines NDR network data with EDR host-level details. Network telemetry shows what is happening within the network. EDR shows what is going on the users' systems. For example, in Incident 1005, NDR alerts detected a port scan, and the fusion engine added EDR data to identify the process (mstsc.exe). The integration of these sources breaks down

information silos. It also provides the LLM's reasoning layer with the context it needs to understand the attacker's intent. As mentioned earlier in section 4.2.2, NDR alerts were mapped with Sysmon events to produce fused alerts. This experiment used static datasets.

5.2.3 Cloud-Based Interpretation and MITRE Mapping

Experimental results demonstrated that cloud models such as Gemini 2.5 Flash performed well at mapping seen behaviours to the MITRE ATT&CK framework. In Incident 1001, for instance, Gemini correctly identified the behaviour as T1021.002 (SMB/Windows Admin Share) and suggested mitigation steps such as isolating affected endpoints and reviewing all authentication, authorisation, and access logs for those systems. It offered detailed results quickly, making it useful for rapid threat analysis. However, using these external APIs can pose data sovereignty risks to enterprise SOC teams tasked with protecting sensitive information.

5.2.4 Local LLM Deployment and Contextual Grounding

The experiment showed that locally hosted models, such as Gemma 3 (27B), can perform almost as well as cloud models. Gemma 3 combined NDR and EDR data to detect lateral movement attempts using stolen credentials. This suggests that organisations with strict privacy requirements can still obtain strong analysis by using high-parameter local models with sufficient context. Smaller models, such as Llama 3.2 (3B), ran faster but often provided generic suggestions rather than leveraging the EDR data. Evaluation and results are presented in Table 5.3, where different trade-offs are seen among models. Gemini offers strong context awareness, a thorough MITRE mapping, and minimal latency ($< 5.0s$). Gemma 3 and Llama 3 have the same detailed analytical capabilities and internal privacy but have substantially higher latency ($8s$). GPT-4o balances with decent performance ($< 3.0s$).

Evaluation Criteria	**GPT-4o**	**Gemini**	**Llama 3**	**Gemma 3**
Context Awareness	High	High	Moderate	High
MITRE Tactic Mapping	N/A	Detailed	N/A	N/A
EDR Correlation Usage	Yes	Yes	Yes	Yes
Operational Privacy	External	External	Internal	Internal
Latency (Seconds)	$< 3.0s$	$< 5 - 6s$	$< 8 - 10s$	$8 - 10s$

Table 5.3: Qualitative Evaluation of LLM Reasoning Output Accuracy and Speed

5.2.5 Privacy-Performance Trade-off Analysis

To run local models, high-performance hardware, particularly GPUs with sufficient VRAM to accommodate large model weights, is required. In business-critical situations, such as Incidents 1003, 1004, etc., involving internal SMB file transfers, the local Gemma 3 model provided more privacy-focused recommendations than cloud models. Cloud models often suggest using third-party external IP lookups. This underscores the importance of local LLMs for organisations in regulated fields such as finance and healthcare.

5.3 Operational Impact: Pseudo-Analyst Workflow Evaluation

To assess the framework's value, a Pseudo-Analyst Evaluation was conducted. This compares the manual steps needed in a typical ML-only workflow with those in the new grounded LLM fusion pipeline for Incident 1001. The results show that the grounded LLM workflow automates the time-consuming, repetitive aspects of incident response. This enables analysts to transition from data miners to decision-makers. Based on the theoretical workflow comparison in Table 5.4, the framework

projects a significant reduction in the time required for manual triage steps. This evaluation also validates research question no. 4, which addressed MTTD/MTTR. However, empirical validation through a live SOC deployment with measured analyst timing will be considered as a future scope of work.

Analyst Task	**Traditional NDR/EDR Workflow**	**Proposed Fusion Pipeline**
Data Aggregation	Manual review of hundreds of logs/alerts.	Single auto-generated summary.
Cross-Tool Correlation	Manual search of EDR and NDR console.	endpoint context included in alert.
Threat Attribution	Rely on personal experience and searching for the mapping.	LLM-provided MITRE mapping.
Investigative Planning	Manual hypothesis generation.	LLM-suggested playbooks.
Total Triage Time	**15–20 Minutes (Assumption)**	**5–10 Minutes (Assumption)**

Table 5.4: Efficiency Gains: Manual Triage vs Proposed Automated Fusion Pipeline

The evaluation of triage efficiency shows the following time improvements, per the assumption according to theoretical and logical concepts:

- **Traditional Workflow (Baseline):** 15–20 minutes per incident.
- **Proposed Fusion Pipeline:** 5–10 minutes per incident.
- **Maximum Time Saved:** A 66.7% reduction in the maximum triage duration, calculated as $\frac{15-5}{15}$.

With the cognitive burden reduced, analysts can focus on proactive threat hunting and strategising comprehensive response actions—furthermore, the standardised output format generated by the remains consistent and less prone to human interpretation errors.

5.4 Evaluation of LLM Hallucination and Narrative Precision

A major operational risk of deploying LLMs in SOC is the generation of false threat intelligence, commonly known as AI hallucination. However, the qualitative evaluation of the incident narratives created in this experiment showed that there was no hallucination. Two main architectural restrictions led to such a narrative precision:

- **Structured Deterministic Ingestion:** Rather than giving the LLM raw, unstructured logs that could overload its context window, the system supplied incidents already aggregated in a strict JSON format. Metrics such as flow durations and packet counts were precomputed, and alerts were subsequently generated, so that the LLM only transformed the fixed data into a narrative. This approach prevented it from making up facts or hallucinating.

- **Strict Prompting and Regex Extraction:** For the generation layer, strictly controlled system prompts, a low temperature setting (temperature = 0.3), and a four-part output (Classification, Narrative, Missing Evidence, and Recommended Steps) were used. The sections needed were retrieved only using Regex. This automated extraction was useful for enforcing the required output structure. However, the factual reliability was evaluated separately by manual comparison with the structured input.

Ultimately, the LLM did not generate any unsupported threat data; instead, it reliably produced high-fidelity narratives that were precisely constrained by the upstream machine-learning and correlation layers.

Six distinct machine-learning-correlated incidents were processed by GPT-4o and Gemini 2.5 Flash, each. These were manually checked against their input schema across five hallucination categories. All structured JSON inputs and corresponding LLM-generated SOC narratives are provided in Appendix B. However, results are shown in Table 5.5, collected from the generated reports in the experiment, for two public LLM models as a sample.

Hallucination Category	**GPT-4o**	**Gemini 2.5 Flash**
Fabricated IP addresses	0/6	0/6
Invented hostnames	0/6	0/6
MITRE IDs not in input	N/A*	0/6
Unsupported severity claims	0/6	0/6
Recommendations contradicting input	0/6	0/6
Total hallucinations / outputs	**0/6**	**0/6**
Hallucination rate	**0%**	**0%**
**Note: GPT-4o is marked as N/A for MITRE ID hallucinations because the model could not generate any MITRE mappings, making an audit for this specific category inapplicable.*		

Table 5.5: Primary Hallucination Audit: Cloud LLMs ($n = 12$ outputs)

In evaluating the narrative precision and operational utility of the generated reports, a significant disparity in tactical depth was observed between the models. While GPT-4o successfully ingested the correlated data to produce coherent, de-

scriptive summaries, it failed to perform advanced threat attribution, yielding zero mappings to the MITRE ATT&CK framework in the table below (Table 5.6) :

Evaluation Metric	**GPT-4o**	**Gemini 2.5 Flash**
MITRE ATT&CK Mapping	0 Techniques Mapped	12 Techniques Mapped (e.g., T1059.001, T1071.004)
Classification Detail	Basic descriptive phrasing	Advanced tactical categorisation
Output Structure	Static paragraph format	Dynamic (Numbered lists for complex actions)
Actionability (DFIR)	Generic SOC investigation steps	Specific forensic actions (Memory dumps, PCAP)
Contextual Depth	Surface-level metric reporting	Deep behavioural analysis (e.g., Living-off-the-Land)

Table 5.6: Comparative Analysis of LLM-Generated Incident Narratives

In contrast, Gemini 2.5 Flash showed more structured contextual reasoning. It clearly linked 12 specific techniques, such as T1059.001 for PowerShell execution and T1071.004 for DNS Tunnelling, to the telemetry data. The recommended steps were also much more actionable. While GPT-4o-mini provided general investigative advice, Gemini 2.5 Flash offered detailed Digital Forensics and Incident Response (DFIR) instructions, such as memory dump collection and tracing parent processes. This shows that although lightweight models can summarise numerical alerts, more advanced models are needed to connect statistical detection with threat intelligence.

Under strict JSON schema constraints, both GPT-4o and Gemini accurately translated fused NDR and EDR telemetry into SOC narratives, with a near-zero hallucination rate, and added no unverified external threat intelligence. These results validate the architectural choice of strict input grounding and show that LLMs require additional output-validation layers for production deployment to reduce hallucination rates.

5.5 Summary of Findings

This end-to-end experiment demonstrates the feasibility of the proposed multi-layered architecture and evaluates its effectiveness. Behavioural ML detection achieved over 95% accuracy, and the correlation layer significantly reduced alert fragmentation, consolidating repeated alerts into incident-level summaries.

Adding grounded LLM reasoning enabled the conversion of large volumes of raw telemetry into clear forensic reports. Comparing different methods shows that combining probabilistic detection, deterministic correlation, and generative reasoning provides advantages over using ML or an LLM alone for lateral movement detection. This tiered architecture also improves context for enterprise security monitoring and reduces the risk of generative hallucinations by grounding the data. Although a comparative LLM performance analysis was conducted to assess deployment trade-offs, the prior goal of this evaluation was not to benchmark generative models. Instead, it shows how the proposed automated SOC pipeline functions throughout the entire process.

6 Conclusion

As modern enterprise networks handle large volumes of telemetry data, it has been challenging to track it all with a single human analyst. Over the years, machine learning models have supported the detection of advanced threats, such as lateral movement. Still, they generate too many fragmented alerts without context, leading to analyst fatigue [8]. This thesis presented a post-detection enhancement framework that links mathematical anomaly detection to real-world human operational response. In the experiment, a clear, strict architectural pipeline was used, in which ML models identify anamalous behaviour. A correlation layer fuses NDR and EDR telemetry data, converting multiple alerts into individual incidents. Finally, a grounded LLM incorporates contextual reasoning. This setup offers a scalable way to reduce the workload of SOC analysts and minimize the risk of hallucinations in cybersecurity.

6.1 Interpretation of Findings

This thesis primarily aimed to assess whether integrating structured contextual correlation with grounded LLM reasoning could enhance post-detection operations in machine learning-driven security.

The experiment demonstrated a substantial operational advantage by isolating detection from reasoning. Traditional ML models such as Isolation Forests and Autoencoders can be used to detect anomalous network behaviour, but their out-

puts are often fragmented and noisy. To correlate NDR alerts with EDR telemetry (Sysmon Event ID 1 and 3), a deterministic correlation layer was implemented to associate network anomalies with specific host processes. This was an important finding as the results showed that feeding the LLM organised, aggregated incident bundles helped to limit hallucinations. Thus, the LLM was an explanation-oriented reasoning layer, rather than a uncontrolled detection component.

The four research questions outlined in Section 1.2 are addressed in the methodological and assessment chapters, as detailed in Table 6.1.

RQ#	Research Question	Sections	Key Findings
RQ1	Operational and interpretability constraints in NDR/EDR systems	§2.7.3, §5.3	While NDR and EDR systems can detect statistical anomalies, they generate a large number of fragmented and unstructured alerts. It makes analysts' workload more difficult to manually correlate isolated alerts across domains to construct coherent threat narratives.
RQ2	How structured contextual correlation reduces alert fragmentation	§3.4, §5.2.1	Deterministic correlation achieved Alert-to-Incident Ratio(AIR) ≈ 95 (575 raw alerts $\rightarrow$ 6 incidents).
RQ3	Balance between false positive reduction and threat priority	§4.4, §5.1	XGBoost achieved F1 = 0.9926; unsupervised models achieved Recall > 0.99 with controlled precision trade-off (0.55).
RQ4	Impact of grounded LLM-assisted reasoning on operational efficiency	§5.3, §5.4	No hallucinations were observed in cloud LLM outputs; theoretical triage time reduction up to 66% (proxy measurement; empirical validation deferred).

Table 6.1: Research Question Mapping to Findings

Operational Implications for the SOC

One significant operational benefit of this architecture is that it helps reduce alert fatigue in the SOC. The network and endpoints generate hundreds of fragmented alerts, which are converted into structured incident summaries. It reduces both the number of alerts and the load on Tier 1 SOC analysts.

Comparing different LLM architectures, such as OpenAI (Cloud), Llama 3.2 3B (Local), and Gemma 3 27B (Local), revealed a key trade-off between forensic detail and response speed. The Gemma 3 model offered very precise forensic ideas but was slow and often timed out. The smaller model, such as Llama 3.2, delivered helpful summaries close to real time. For SOC deployment, the appropriate model choice depends on the balance between privacy, speed, reasoning depth and available computational resources.

6.2 Limitations of the thesis

Even though the proposed architecture showed promising results, several important limitations should be acknowledged:

- **Simulated Telemetry Fusion and Synthetic Correlation:** Perfectly synchronised, high-quality datasets containing both NDR and EDR telemetry for identical attack vectors are exceptionally rare in public academic repositories. To evaluate the pipeline's cross-domain reasoning, this study utilised two independent datasets (CIC-IDS-2018 and Empire APT3 Sysmon). Because these datasets were not recorded on the same physical infrastructure, aligning alerts solely by a timestamp window is insufficient; in a high-volume telemetry environment, relying only on time could result in the coincidental fusion of unrelated network and endpoint attacks. In a real-world enterprise environment, this limitation can be mitigated if NDR and EDR systems share identifiers,

such as matching IP addresses, MAC addresses, and active user sessions. This makes it possible to combine their data accurately, without needing artificial behavioural mapping.

- **Absence of Human-in-the-Loop Testing:** The fusion of NDR-EDR alerts and alert compression ratios was used as a proxy to figure out how much the MTTR was reduced. However, the experiment could not test the system with live SOC analysts to assess how it would affect human trust or triage speed. This remains a complex factor in Human-AI collaboration.

- **Dependence on Traditional Machine Learning instead of Deep Learning:** Modern security architectures (Next-Generation Firewalls, modern WAFs, EDR and commercial XDR platforms) are adopting deep learning (DL) architectures. On the other hand, this research was explicitly limited to traditional Machine Learning algorithms (e.g., Random Forest, XGBoost). Traditional ML needs deliberate, domain-specific feature engineering, such as directly extracting Cmdline_Length or counting Hidden_Flags from Sysmon logs. While very efficient and interpretable, this approach may not capture highly complex, non-linear, or temporally obfuscated attack patterns that a Deep Neural Network could, in principle, learn from raw, unparsed log data.

- **Explainability Trade-off (XAI):** The thesis was deliberately targeted at the model explainability rather than sophisticated deep architectural design. Tier-3 SOC analysts and the automated LLM interpretation layer must explicitly understand why an alert triggered (e.g., "Port 445 Volume" is a feature that triggered the NDR alerts). It is very difficult to extract the exact parameters needed for the fusion and contextualisation layers without complex eXplainable AI (XAI) frameworks, which were beyond the scope of this thesis.

6.3 Future Research Directions

The findings of this thesis open several promising avenues for future research and development:

- **Retrieval-Augmented Generation (RAG) for Threat Intelligence:** In future, this architecture could include an RAG framework. If the LLM can query live cyber threat intelligence (CTI) databases or internal corporate playbooks, the reasoning layer could automatically incorporate the latest IoCs and the organisation's own mitigation steps into its responses.

- **Deep Learning Integration for Automated Feature Extraction:** Future versions of this cross-domain fusion pipeline should consider the integration of Deep Learning models such as Convolutional Neural Networks (CNNs) or deep Autoencoders. These architectures can be trained to ingest raw packet bytes and raw endpoint event streams, rather than relying on manual feature engineering from PCAP and EDR JSON files. This would enable the detection layer to automatically learn latent representations of emerging cyber threats, potentially detecting zero-day attack vectors that are missed by typical ML feature sets [87].

- **Empirical SOC Deployment:** A long-term study in a mid-sized enterprise SOC would give valuable data on how human analysts interact with LLM-generated narratives over time. Most importantly, when it comes to building or losing trust in AI outputs.

- **Experiment with Live and larger datasets:** To achieve a better evaluation in terms of MTTD/MTTR from the SOC perspective, future research can assess the framework utilising more extensive datasets, preferably real-time or near-real-time telemetry streams. The present experiment employs controlled

public datasets and simulated endpoint enrichment to support reproducibility. This is a good approach for a first-stage evaluation. Still, real enterprise telemetry would enable SOC analysts to investigate concept drift, changing attacker behaviour, cross-source inconsistencies, and deployment lag more realistically. A streaming implementation utilising technologies such as Apache Kafka, Splunk, Elastic, or Microsoft Sentinel could enable direct measurement of end-to-end operational times, rather than relying on proxy indicators.

This research focused on establishing a foundational framework for fusing NDR and EDR telemetry and validating its contextual output. To establish a highly reliable baseline for automated narrative generation, the evaluation phase was deliberately scoped to utilise state-of-the-art cloud APIs (GPT-4o, Gemini). While enterprise SOCs frequently require localised, privacy-preserving models, a comprehensive quantitative benchmarking of on-premises LLMs (such as Llama 3 or Gemma 3) falls outside the scope of this thesis. Future research should build upon the cloud-validated baseline established in this study to quantify the specific computational and accuracy trade-offs required for fully air-gapped SOC deployments.

- **Commercial XDR Deep Learning Interoperability:** Commercial security vendors such as CrowdStrike, SentinelOne, and Trend Micro apply proprietary deep neural network models at the edge. Further research is needed to determine effective methods for integrating opaque, or "black-box," outputs into comprehensive detection workflows. This research could improve interoperability and detection accuracy by informing a unified mathematical framework that integrates deep learning confidence scores with traditional machine learning outputs, thereby facilitating more effective vendor-agnostic XDR environments.

References

[1] A. Sperotto, G. Schaffrath, R. Sadre, C. Morariu, A. Pras, and B. Stiller, "An Overview of IP Flow-Based Intrusion Detection", *IEEE Communications Surveys & Tutorials*, vol. 12, no. 3, pp. 343–356, 2010. DOI: 10.1109/SURV.2010.032210.00054.

[2] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly Detection: A Survey", *ACM Computing Surveys*, vol. 41, no. 3, 15:1–15:58, 2009. DOI: 10.1145/1541880.1541882.

[3] M. Ring, "A Survey of Network-Based Intrusion Detection Data Sets", *Computers & Security*, vol. 86, pp. 147–167, 2019. DOI: 10.1016/j.cose.2019.06.005.

[4] T. Campfield, "Network Detection and Response: Closing the Visibility Gap", *Network Security*, vol. 2020, no. 9, pp. 18–20, 2020. DOI: 10.1016/S1353-4858(20)30104-5.

[5] Srinivas, "AI-Augmented SOC: A Survey of LLMs and Agents for Security Automation", *Journal of Cybersecurity and Privacy*, vol. 5, no. 4, p. 95, 2025. DOI: 10.3390/jcp5040095.

[6] L. Huang, "Survey on Hallucination in Large Language Models", *arXiv*, 2023. DOI: 10.48550/arXiv.2311.05232.

[7] S. C. Sundaramurthy, A. G. Bardas, J. Case, X. Ou, M. Wesch, J. McHugh, and S. R. Rajagopalan, "A Human Capital Model for Mitigating Security Analyst Burnout", in *Eleventh Symposium On Usable Privacy and Security (SOUPS 2015)*, Accessed: Jun. 5, 2026, Ottawa, ON, Canada, 2015, pp. 347–359. [Online]. Available: https://www.usenix.org/system/files/conference/soups2015/soups15-paper-sundaramurthy.pdf.

[8] S. Tariq, M. B. Chhetri, S. Nepal, and C. Paris, "Alert Fatigue in Security Operations Centres: Research Challenges and Opportunities", *ACM Computing Surveys*, vol. 57, pp. 1–38, 2025. DOI: 10.1145/3723158.

[9] MITRE Corporation, *MITRE ATT&CK: Lateral Movement*, [Online]. Available: https://attack.mitre.org/tactics/TA0008/. Accessed: May. 30, 2026.

[10] C. Smiliotopoulos, G. Bendiab, S. Shiaeles, and N. Savage, "Detecting Lateral Movement: A Systematic Survey", *Heliyon*, vol. 10, no. 7, e26317, 2024. DOI: 10.1016/j.heliyon.2024.e26317.

[11] F. Barr-Smith, X. Ugarte-Pedrero, M. Graziano, R. Spolaor, and I. Martinovic, "Survivalism: Systematic Analysis of Windows Malware Living-Off-The-Land", in *2021 IEEE Symposium on Security and Privacy (SP)*, Virtual Event: IEEE, 2021, pp. 1557–1574. DOI: 10.1109/SP40001.2021.00047.

[12] A. Alsaheel, Y. Nan, S. Ma, L. Yu, G. Walkup, Z. B. Celik, X. Zhang, and D. Xu, "ATLAS: A Sequence-based Learning Approach for Attack Investigation", in *30th USENIX Security Symposium (USENIX Security 21)*, Accessed: Jun. 5, 2026, Virtual Event: USENIX Association, 2021, pp. 3005–3022. [Online]. Available: https://www.usenix.org/conference/usenixsecurity21/presentation/alsaheel.

[13] F. Liu, Y. Wen, D. Zhang, X. Jiang, X. Xing, and D. Meng, "Log2vec: A Heterogeneous Graph Embedding Based Approach for Detecting Cyber Threats within Enterprise", in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, London, UK: ACM, 2019, pp. 1777–1794. DOI: 10.1145/3319535.3363224.

[14] E. Papadogiannaki and S. Ioannidis, "A Survey on Encrypted Network Traffic Analysis Applications, Techniques, and Countermeasures", *ACM Computing Surveys*, vol. 54, no. 6, pp. 1–35, 2021. DOI: 10.1145/3457904.

[15] J. Navarro, A. Deruyver, and P. Parrend, "A Systematic Survey on Multi-step Attack Detection", *Computers & Security*, vol. 76, pp. 214–249, 2018. DOI: 10.1016/j.cose.2018.03.001.

[16] H. Hindy, D. Brosset, E. Bayne, A. K. Seeam, C. Tachtatzis, R. Atkinson, and M. Bures, "A Taxonomy of Network Threats and the Effect of Current Datasets on Intrusion Detection Systems", *IEEE Access*, vol. 8, pp. 104 650–104 675, 2020. DOI: 10.1109/ACCESS.2020.3000179.

[17] S. Homayoun, M. Haraldsdóttir, E. Lynge, and C. D. Jensen, "Effective Noise Reduction in NDR Systems: A Simple Yet Powerful Apriori-Based Approach", *Sensors*, vol. 24, no. 20, p. 6547, 2024. DOI: 10.3390/s24206547.

[18] A. Alshamrani, S. Myneni, A. Chowdhary, and D. Huang, "A Survey on Advanced Persistent Threats: Techniques, Solutions, Challenges, and Research Opportunities", *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1851–1877, 2019. DOI: 10.1109/COMST.2019.2891891.

[19] S. M. Milajerdi, R. Ghasemiesfeh, B. Eshete, R. Sekar, and S. J. Stolfo, "HOLMES: Real-time APT Detection through Correlation of Suspicious Information Flows", in *2019 IEEE Symposium on Security and Privacy (SP)*,

San Francisco, CA, USA, 2019, pp. 1137–1152. DOI: 10.48550/arXiv.1810.01594.

[20] W. U. Hassan, A. Bates, and D. Marino, "Tactical Provenance Analysis for Endpoint Detection and Response Systems", in *2020 IEEE Symposium on Security and Privacy (SP)*, San Francisco, CA, USA: IEEE, 2020, pp. 1172–1189. DOI: 10.1109/SP40000.2020.00096.

[21] B. Zhang, Y. Gao, B. Kuang, C. Yu, A. Fu, and W. Susilo, "A Survey on Advanced Persistent Threat Detection: A Unified Framework, Challenges, and Countermeasures", *ACM Computing Surveys*, vol. 56, no. 4, pp. 1–36, 2024. DOI: 10.1145/3700749.

[22] Z. Ahmad, A. S. Khan, C. W. Shiang, J. Abdullah, and F. Ahmad, "Network Intrusion Detection System: A Systematic Study of Machine Learning and Deep Learning Approaches", *Transactions on Emerging Telecommunications Technologies*, vol. 32, no. 1, e4150, 2021. DOI: 10.1002/ett.4150.

[23] A. L. Buczak and E. Guven, "A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection", *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1153–1176, 2016. DOI: 10.1109/COMST.2015.2494502.

[24] M. Cantone, C. Marrocco, and A. Bria, "Machine Learning in Network Intrusion Detection: A Cross-Dataset Generalization Study", *IEEE Access*, vol. 12, pp. 144 489–144 508, 2024. DOI: 10.1109/ACCESS.2024.3472907.

[25] S. S. Karim, M. Afzal, W. Iqbal, and D. Al Abri, "Advanced Persistent Threat (APT) and Intrusion Detection Evaluation Dataset for Linux Systems 2024", *Data in Brief*, vol. 54, p. 110 290, 2024. DOI: 10.1016/j.dib.2024.110290.

[26] Z. Weng, W. Zhang, T. Zhu, Z. Dou, H. Sun, Z. Ye, and Y. Tian, "RT-APT: A Real-Time APT Anomaly Detection Method for Large-Scale Provenance

Graph", *Journal of Network and Computer Applications*, vol. 233, p. 104 036, 2025. DOI: 10.1016/j.jnca.2024.104036.

[27] Z. Cheng, Q. Lv, J. Liang, Y. Wang, D. Sun, T. Pasquier, and X. Han, "Kairos: Provenance-Based Intrusion Detection", in *2024 IEEE Symposium on Security and Privacy (SP)*, San Francisco, CA, USA: IEEE, 2024, pp. 5–22. DOI: 10.1109/SP54263.2024.00005.

[28] M. U. Rehman, H. Ahmadi, and W. U. Hassan, "Flash: Intrusion Detection via Provenance Graph Learning", in *2024 IEEE Symposium on Security and Privacy (SP)*, San Francisco, CA, USA: IEEE, 2024, pp. 139–157. DOI: 10.1109/SP54263.2024.00139.

[29] C. Dong, J. Yang, S. Liu, Z. Wang, Y. Liu, and Z. Lu, "Behavior Deviation Measurement for Lateral Movement", *Computers & Security*, vol. 128, p. 103 267, 2023. DOI: 10.1016/j.cose.2023.103267.

[30] A. Habibzadeh, F. Feyzi, and R. E. Atani, "Large Language Models for Security Operations Centers: A Survey", *arXiv preprint arXiv:2509.10858*, 2025, Accessed: Jun. 5, 2026. [Online]. Available: https://arxiv.org/abs/2509.10858.

[31] Y. Yigit et al., "Review of Generative AI Methods in Cybersecurity", *arXiv preprint arXiv:2403.08701*, 2024, Accessed: Jun. 5, 2026. [Online]. Available: https://arxiv.org/abs/2403.08701.

[32] M. Turcotte, F. Labrèche, and S.-O. Paquette, "Automated Alert Classification and Triage (AACT)", *arXiv preprint arXiv:2505.09843*, 2025, Accessed: Jun. 5, 2026. [Online]. Available: https://arxiv.org/abs/2505.09843.

[33] O. Oniagbi, "Evaluation of LLM Agents for the SOC Tier 1 Analyst Triage Process", Accessed: May 30, 2026, Master's Thesis, University of Turku, 2024. [Online]. Available: https://www.utupub.fi/handle/10024/178601.

[34] Z. Hu, L. Liu, H. Li, C. Song, M. Ge, Q. Guo, L. Ma, and X. Yu, "CCLog: Actionable APT Forensics via Fused Log Semantics and Provenance Graph Topology", *Computer Networks*, vol. 272, p. 111 660, 2025. DOI: 10.1016/j.comnet.2025.111660.

[35] I. J. King and H. H. Huang, "EULER: Detecting Network Lateral Movement via Temporal Graph Link Prediction", in *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, USA: Internet Society, 2022. DOI: 10.14722/ndss.2022.24107.

[36] M. Noppel and C. Wressnegger, "SoK: Explainable Machine Learning in Adversarial Environments", in *2024 IEEE Symposium on Security and Privacy (SP)*, San Francisco, CA, USA: IEEE, 2024, pp. 21–38. DOI: 10.1109/SP54263.2024.00021.

[37] L. Sadlek, M. M. Yamin, P. Čeleda, and B. Katt, "Severity-Based SOC Triage Using Kill Chain Graphs", *Journal of Information Security and Applications*, vol. 82, p. 103 956, 2025. DOI: 10.1016/j.jisa.2024.103956.

[38] Z. Ji, N. Lee, R. M. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y. J. Bang, A. Madotto, and P. Fung, "Survey of Hallucination in Large Language Models", *ACM Computing Surveys*, vol. 55, no. 12, pp. 1–38, 2023. DOI: 10.1145/3571730.

[39] N. Capuano, G. Fenza, V. Loia, and C. Stanzione, "Explainable Artificial Intelligence in Cybersecurity: A Survey", *IEEE Access*, vol. 10, pp. 9877–9890, 2022. DOI: 10.1109/ACCESS.2022.3204171.

[40] OWASP Foundation, *LLM02:2025 Sensitive Information Disclosure*, Accessed: Jun. 5, 2026, 2025. [Online]. Available: https://genai.owasp.org/llmrisk/llm022025-sensitive-information-disclosure/.

[41] T. D. Wagner, K. Mahbub, E. Palomar, and A. E. Abdallah, "Cyber Threat Intelligence Sharing: Survey and Research Directions", *Computers & Security*, vol. 87, p. 101 589, 2019. DOI: 10.1016/j.cose.2019.101589.

[42] M. Hassanin and N. Moustafa, "A Comprehensive Overview of Large Language Models (LLMs) for Cyber Defences: Opportunities and Directions", *arXiv preprint arXiv:2405.14487*, 2024, Accessed: Jun. 5, 2026. [Online]. Available: https://arxiv.org/abs/2405.14487.

[43] JPCERT Coordination Center, "Detecting Lateral Movement through Tracking Event Logs", JPCERT Coordination Center, Research Report, 2017, Accessed: Jun. 5, 2026. [Online]. Available: https://www.jpcert.or.jp/english/pub/sr/DetectingLateralMovementThroughTrackingEventLogs_version2.pdf.

[44] T. T. T. Nguyen and G. Armitage, "A Survey of Techniques for Internet Traffic Classification and Classification with Encryption", *IEEE Communications Surveys & Tutorials*, vol. 10, no. 4, pp. 56–76, 2008. DOI: 10.1109/SURV.2008.080406.

[45] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A Detailed Analysis of the KDD CUP 99 Data Set", in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, Ottawa, ON, Canada, 2009, pp. 1–6. DOI: 10.1109/CISDA.2009.5356528.

[46] European Union, *Directive (EU) 2022/2555 (NIS 2 Directive)*, 2022.

[47] R. M. Achmad, D. P. Nariswari, B. A. Pratomo, and H. Studiawan, "Sysmon Event Logs for Machine Learning-Based Malware Detection", *Cyber Security and Applications*, vol. 3, p. 100 110, 2025. DOI: 10.1016/j.csa.2025.100110.

[48] B. Bowman, C. Huang, J. S. Moore, and B. B. Miller, "Detecting Lateral Movement in Enterprise Computer Networks with Unsupervised Graph Learn-

ing", in *23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020)*, Accessed: May. 30, 2026, Virtual Event: USENIX Association, 2020, pp. 257–273. [Online]. Available: https://www.usenix.org/conference/raid2020/presentation/bowman.

[49] D. Kramer, L. Rosique, A. Narotam, E. Bursztein, P. G. Kelley, K. Thomas, and A. Woodruff, "Integrating Large Language Models into Security Incident Response", in *Proceedings of the Twenty-First Symposium on Usable Privacy and Security (SOUPS 2025)*, Seattle, WA, USA: USENIX Association, Aug. 2025, ISBN: 978-1-939133-51-9. [Online]. Available: https://www.usenix.org/conference/soups2025/presentation/kramer.

[50] R. Singh, S. Tariq, F. Jalalvand, M. Baruwal Chhetri, S. Nepal, C. Paris, and M. Lochner, "LLMs in the SOC: An Empirical Study of Human-AI Collaboration in Security Operations Centres", *arXiv preprint arXiv:2508.18947*, 2025. [Online]. Available: https://arxiv.org/abs/2508.18947.

[51] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. K"uttler, M. Lewis, W.-t. Yih, T. Rockt"aschel, S. Riedel, and D. Kiela, "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks", in *Advances in Neural Information Processing Systems (NeurIPS)*, Accessed: Jun. 5, 2026, vol. 33, Virtual Event, 2020, pp. 9459–9470. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html.

[52] S. Prakash, S. A. Mary, G. Sudhagar, and M. Batumalay, "Hybrid Ensemble Learning for Anomaly Detection in Metaverse Transactions Using Isolation Forest, Autoencoder, and XGBoost", *International Journal Research on Metaverse*, vol. 3, no. 1, pp. 64–80, 2026. DOI: 10.47738/ijrm.v3i1.46.

[53] K. Stroeh, E. R. M. Madeira, and S. K. Goldenstein, "An Approach to the Correlation of Security Events Based on Machine Learning Techniques", *Journal of Internet Services and Applications*, vol. 4, no. 1, p. 7, 2013. DOI: 10.1186/1869-0238-4-7.

[54] S. Kotilingala, "Next-Gen SOC: Leveraging Generative AI for Scalable Threat Detection and AI-Powered Alert Classification", *International Journal of Applied Mathematics, Sciences, and Technology for National Defense*, vol. 3, no. 3, pp. 143–152, 2025. DOI: 10.58524/app.sci.def.v3i3.670.

[55] E. Li, T. Mallick, E. Rose, W. Robertson, A. Oprea, and C. Nita-Rotaru, "ACE: A Security Architecture for LLM-Integrated App Systems", in *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, Accessed: Jun. 5, 2026, San Diego, CA, USA, 2026. [Online]. Available: https://www.ndss-symposium.org/ndss-paper/ace-a-security-architecture-for-llm-integrated-app-systems/.

[56] I. Kotenko, D. Gaifulina, and I. Zelichenok, "Systematic Literature Review of Security Event Correlation Methods", *IEEE Access*, vol. 10, pp. 43 387–43 393, 2022. DOI: 10.1109/ACCESS.2022.3168976.

[57] R. Kaur, D. Gabrijelčič, and T. Klobučar, "Artificial Intelligence for Cybersecurity: Literature Review and Future Research Directions", *Information Fusion*, vol. 97, p. 101 804, 2023. DOI: 10.1016/j.inffus.2023.101804.

[58] A. Mahboubi, K. Luong, H. Aboutorab, H. T. Bui, G. Jarrad, M. Bahutair, S. Camtepe, G. Pogrebna, E. Ahmed, B. Barry, and H. Gately, "Evolving Techniques in Cyber Threat Hunting: A Systematic Review", *Journal of Network and Computer Applications*, vol. 232, p. 104 004, 2024. DOI: 10.1016/j.jnca.2024.104004.

[59] P. Garcia-Teodoro, J. Diaz-Verdejo, G. Macia-Fernandez, and E. Vazquez, "Anomaly-Based Network Intrusion Detection: Techniques, Systems and Challenges", *Computers & Security*, vol. 28, no. 1–2, pp. 18–28, 2009. DOI: 10.1016/j.cose.2008.08.003.

[60] G. Pang, C. Shen, L. Cao, and A. van den Hengel, "Deep Learning for Anomaly Detection: A Review", *ACM Computing Surveys*, vol. 54, no. 2, pp. 1–38, 2021. DOI: 10.1145/3439950.

[61] J. Kreps, N. Narkhede, and J. Rao, "Kafka: A Distributed Messaging System for Log Processing", in *Proceedings of the NetDB Workshop*, Accessed: Jun. 5, 2026, Athens, Greece, 2011. [Online]. Available: https://notes.stephenholiday.com/Kafka.pdf.

[62] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System", in *Proceedings of the ACM SIGKDD*, San Francisco, CA, USA, 2016, pp. 785–794. DOI: 10.1145/2939672.2939785.

[63] M. Ahmed, A. N. Mahmood, and J. Hu, "A Survey of Network Anomaly Detection Techniques", *Journal of Network and Computer Applications*, vol. 60, pp. 19–31, 2016. DOI: 10.1016/j.jnca.2015.11.016.

[64] C. Merlano, "Enhancing Cyber Security through Artificial Intelligence and Machine Learning: A Literature Review", *Journal of Cyber Security*, vol. 6, no. 1, pp. 89–116, 2024. DOI: 10.32604/jcs.2024.056164.

[65] M. Husák, J. Komárková, E. Bou-Harb, and P. Čeleda, "Survey of Attack Projection, Prediction, and Forecasting in Cyber Security", *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 640–660, 2019. DOI: 10.1109/COMST.2018.2871866.

[66] F. Jalalvand, M. Baruwal Chhetri, S. Nepal, and C. Paris, "Alert Prioritisation in Security Operations Centres: A Systematic Survey on Criteria and

Methods", *ACM Computing Surveys*, vol. 57, no. 2, 42:1–42:36, 2024. DOI: 10.1145/3695462.

[67] R. Sommer and V. Paxson, "Outside the Closed World: On Using Machine Learning for Network Intrusion Detection", in *IEEE Symposium on Security and Privacy*, Oakland, CA, USA, 2010, pp. 305–316. DOI: 10.1109/SP.2010.25.

[68] J. B. D. Cabrera, L. Lewis, and R. Mehra, "Detection and Classification of Intrusions and Faults Using Sequences of System Calls", *ACM SIGMOD Record*, vol. 30, no. 4, pp. 25–34, 2001. DOI: 10.1145/604264.604269.

[69] L. Breiman, "Random Forests", *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001. DOI: 10.1023/A:1010933404324.

[70] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation Forest", in *IEEE International Conference on Data Mining (ICDM)*, Pisa, Italy, 2008, pp. 413–422. DOI: 10.1109/ICDM.2008.17.

[71] M. Sakurada and T. Yairi, "Anomaly Detection Using Autoencoders with Nonlinear Dimensionality Reduction", in *Proceedings of the MLSDA Workshop (ACM)*, New York, NY, USA, 2014. DOI: 10.1145/2689746.2689747.

[72] K. Julisch, "Clustering Intrusion Detection Alarms to Support Root Cause Analysis", *ACM Transactions on Information and System Security*, vol. 6, no. 4, pp. 443–471, 2003. DOI: 10.1145/950191.950192.

[73] P. Ning, Y. Cui, and D. Reeves, "Constructing Attack Scenarios through Correlation of Intrusion Alerts", in *Proceedings of the ACM CCS*, Washington, DC, USA, 2002, pp. 245–254. DOI: 10.1145/586110.586144.

[74] Stanford Center for Research on Foundation Models, "On the Opportunities and Risks of Foundation Models", *arXiv preprint arXiv:2108.07258*, 2021, Ac-

cessed: Jun. 5, 2026. [Online]. Available: https://arxiv.org/abs/2108.07258.

[75] K. Scarfone and P. Mell, *Guide to Intrusion Detection and Prevention Systems (IDPS)*, Accessed: May. 30, 2026, 2007. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-94.pdf.

[76] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. Le, and D. Zhou, "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models", *Advances in Neural Information Processing Systems*, vol. 35, pp. 24 824–24 837, 2022, Accessed: Jun. 5, 2026. [Online]. Available: https://arxiv.org/abs/2201.11903.

[77] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization", in *ICISSP*, Funchal, Madeira, Portugal, 2018, pp. 108–116. DOI: 10.5220/0006639801080116.

[78] Canadian Institute for Cybersecurity, *CSE-CIC-IDS2018 Dataset*, Accessed: Jun. 5, 2026, 2018. [Online]. Available: https://www.unb.ca/cic/datasets/ids-2018.html.

[79] Microsoft, *Sysmon (System Monitor)*, Accessed: Jun. 5, 2026, 2022. [Online]. Available: https://learn.microsoft.com/en-us/sysinternals/downloads/sysmon.

[80] H. Maosa, K. Ouazzane, and M. C. Ghanem, "A Hierarchical Security Event Correlation Model for Real-Time Threat Detection and Response", *Network*, vol. 4, no. 1, pp. 68–90, 2024. DOI: 10.3390/network4010004.

[81] Z. Chen, J. Chen, A. Singh, and M. Sra, "XplainLLM: A Knowledge-Augmented Dataset for Reliable Grounded Explanations in LLMs", *arXiv preprint*, 2023,

Accessed: Jun. 5, 2026. [Online]. Available: https://arxiv.org/abs/2311.08614.

[82] H. He and E. A. Garcia, "Learning from Imbalanced Data", *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, 2009. DOI: 10.1109/TKDE.2008.239.

[83] M. Sokolova and G. Lapalme, "A Systematic Analysis of Performance Measures for Classification Tasks", *Information Processing & Management*, vol. 45, no. 4, pp. 427–437, 2009. DOI: 10.1016/j.ipm.2009.03.002.

[84] D. M. W. Powers, "Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness and Correlation", *Journal of Machine Learning Technologies*, vol. 2, no. 1, pp. 37–63, 2011, Accessed: Jun. 5, 2026. [Online]. Available: https://arxiv.org/abs/2010.16061.

[85] M. Landauer, F. Skopik, M. Wurzenberger, and A. Rauber, "Dealing with Security Alert Flooding: Using Machine Learning for Domain-Independent Alert Aggregation", *ACM Transactions on Privacy and Security*, vol. 25, no. 3, pp. 1–36, 2022. DOI: 10.1145/3510581.

[86] M. Lyu, H. H. Gharakheili, and V. Sivaraman, "A Survey on Enterprise Network Security: Asset Behavioral Monitoring and Distributed Attack Detection", *IEEE Access*, vol. 12, pp. 89 363–89 377, 2024. DOI: 10.1109/ACCESS.2024.3419068.

[87] M. J. Ahmed, "Enhancing Explainability and Performance in Intrusion Detection Systems using Deep Learning Models and LLMs", Accessed: May. 30, 2026, M.S. thesis, University of Turku, 2025. [Online]. Available: https://urn.fi/URN:NBN:fi-fe2025080881613.

[88] S. Yang, X. Zheng, X. Zhang, J. Xu, J. Li, D. Xie, W. Long, and E. C. H. Ngai, "Large Language Models for Network Intrusion Detection Systems: Founda-

tions, Implementations, and Future Directions", *arXiv preprint arXiv:2507.04752*, 2025, Accessed: Jun. 5, 2026. [Online]. Available: https://arxiv.org/abs/2507.04752.

Appendix A Selected Implementation Code

A.1 Dataset Training Sample

```
[*] STEP 1: Loading NDR/Lateral Movement Data...
    -> Successfully loaded a 100000 row sample.
Columns in dataset: ['Protocol', 'Flow Duration', 'Total Fwd Packets', 'Total Backward Packets', 'Fwd Packets Length Total', 'Bwd Packets Length Tota
l', 'Fwd Packet Length Max', 'Fwd Packet Length Min', 'Fwd Packet Length Mean', 'Fwd Packet Length Std', 'Bwd Packet Length Max', 'Bwd Packet Length M
in', 'Bwd Packet Length Mean', 'Bwd Packet Length Std', 'Flow Bytes/s', 'Flow Packets/s', 'Flow IAT Mean', 'Flow IAT Std', 'Flow IAT Max', 'Flow IAT M
in', 'Fwd IAT Total', 'Fwd IAT Mean', 'Fwd IAT Std', 'Fwd IAT Max', 'Fwd IAT Min', 'Bwd IAT Total', 'Bwd IAT Mean', 'Bwd IAT Std', 'Bwd IAT Max', 'Bwd
IAT Min', 'Fwd PSH Flags', 'Bwd PSH Flags', 'Fwd URG Flags', 'Bwd URG Flags', 'Fwd Header Length', 'Bwd Header Length', 'Fwd Packets/s', 'Bwd Packets/
s', 'Packet Length Min', 'Packet Length Max', 'Packet Length Mean', 'Packet Length Std', 'Packet Length Variance', 'FIN Flag Count', 'SYN Flag Count',
'RST Flag Count', 'PSH Flag Count', 'ACK Flag Count', 'URG Flag Count', 'CWE Flag Count', 'ECE Flag Count', 'Down/Up Ratio', 'Avg Packet Size', 'Avg F
wd Segment Size', 'Avg Bwd Segment Size', 'Fwd Avg Bytes/Bulk', 'Fwd Avg Packets/Bulk', 'Fwd Avg Bulk Rate', 'Bwd Avg Bytes/Bulk', 'Bwd Avg Packets/Bu
lk', 'Bwd Avg Bulk Rate', 'Subflow Fwd Packets', 'Subflow Fwd Bytes', 'Subflow Bwd Packets', 'Subflow Bwd Bytes', 'Init Fwd Win Bytes', 'Init Bwd Win
Bytes', 'Fwd Act Data Packets', 'Fwd Seg Size Min', 'Active Mean', 'Active Std', 'Active Max', 'Active Min', 'Idle Mean', 'Idle Std', 'Idle Max', 'Idl
e Min', 'Label']
Selected behavioral features: ['Flow Duration', 'Total Fwd Packets', 'Total Backward Packets', 'Fwd Packets Length Total', 'Bwd Packets Length Total',
'Fwd Packet Length Max', 'Fwd Packet Length Min', 'Fwd Packet Length Mean', 'Fwd Packet Length Std', 'Bwd Packet Length Max', 'Bwd Packet Length Min',
'Bwd Packet Length Mean', 'Bwd Packet Length Std', 'Flow Packets/s', 'Flow IAT Mean', 'Flow IAT Std', 'Flow IAT Max', 'Flow IAT Min', 'Fwd IAT Total',
'Fwd IAT Mean', 'Fwd IAT Std', 'Fwd IAT Max', 'Fwd IAT Min', 'Bwd IAT Total', 'Bwd IAT Mean', 'Bwd IAT Std', 'Bwd IAT Max', 'Bwd IAT Min', 'Fwd PSH Fl
ags', 'Bwd PSH Flags', 'Fwd URG Flags', 'Bwd URG Flags', 'Fwd Header Length', 'Bwd Header Length', 'Fwd Packets/s', 'Bwd Packets/s', 'Packet Length Mi
n', 'Packet Length Max', 'Packet Length Mean', 'Packet Length Std', 'Packet Length Variance', 'FIN Flag Count', 'SYN Flag Count', 'RST Flag Count', 'P
SH Flag Count', 'ACK Flag Count', 'URG Flag Count', 'CWE Flag Count', 'ECE Flag Count', 'Down/Up Ratio', 'Avg Packet Size', 'Avg Fwd Segment Size', 'A
vg Bwd Segment Size', 'Fwd Avg Packets/Bulk', 'Bwd Avg Packets/Bulk', 'Subflow Fwd Packets', 'Subflow Bwd Packets', 'Fwd Act Data Packets', 'Fwd Seg S
ize Min', 'Active Mean', 'Active Std', 'Active Max', 'Active Min']
    -> Scaled and ready. Training on 63 strictly behavioral features.
```

Figure A.1: Sample console output showing dataset loading, feature selection, and training preparation for the NDR lateral movement dataset.

A.2 NDR and EDR Alert Generation

```
1 # NDR alert generation using XGBoost probability threshold
2 df_ndr["NDR_Alert"] = (
3     xgb.predict_proba(X_ndr_scaled)[:, 1] >= 0.12
```

```
).astype(int)

# EDR alert generation using trained endpoint model
edr_model = RandomForestClassifier(
    n_estimators=25,
    max_depth=3,
    random_state=42
)

edr_model.fit(X_edr, y_edr)

final_edr_df["EDR_Alert"] = edr_model.predict(X_edr)
```

Listing A.1: NDR and EDR alert generation

A.3 Automated Temporal Alignment of Disparate Datasets

```
# 1. Abstracting time into discrete automated windows for the
   NDR dataset
df_ndr["Time_Window"] = (
    np.arange(len(df_ndr)) // np.random.randint(80, 120, size=
       len(df_ndr))
)
attack_windows = df_ndr[df_ndr["Is_Attack"] == 1]["Time_Window
   "].unique()

# 2. Initializing EDR temporal dataframe with baseline noise
aligned_edr_records = []
```

```
for window in df_ndr["Time_Window"].unique():
    record = {col: 0 for col in edr_features.columns}
    record["Time_Window"] = window
    record["Avg_Cmdline_Length"] = np.random.randint(10, 45)
    record["Total_Sysmon_Events"] = np.random.randint(10, 50)
    aligned_edr_records.append(record)

final_edr_df = pd.DataFrame(aligned_edr_records).set_index("
   Time_Window")

# 3. Algorithmically mapping the parsed Mordor features (
   Sysmon) into the temporal grid
mordor_index = 0
for window in attack_windows:
    if mordor_index < len(edr_features):
        for col in edr_features.columns:
            final_edr_df.loc[window, col] = edr_features.iloc[
               mordor_index][col]
        mordor_index += 1

final_edr_df.reset_index(inplace=True)
```

Listing A.2: Establishing a shared temporal grid for automated mapping

A.4 Automated Fusion of NDR and EDR Alerts

```
# 1. THE AUTOMATED TEMPORAL JOIN
# Merge independent NDR and EDR outputs using the shared
   Time_Window key.
```

```
# This algorithmically aligns the two datasets without manual mapping.
fused_windows = pd.merge(
    df_ndr.groupby("Time_Window")["NDR_Alert"].max().reset_index(),
    final_edr_df,
    on="Time_Window"
)

# 2. THE AUTOMATED 'AND' GATE
# Dynamically extract only the windows where both ML models independently
# flag an anomaly simultaneously, effectively filtering false positives.
true_fused_windows = fused_windows[
    (fused_windows["NDR_Alert"] == 1) &
    (fused_windows["EDR_Alert"] == 1)
]["Time_Window"].unique()

# Select top confident windows for LLM processing
top_windows = (
    true_fused_windows[:6]
    if len(true_fused_windows) >= 6
    else true_fused_windows
)

# Fallback mechanism if no synchronised fused window is found.
if len(top_windows) == 0:
    top_windows = (
```

```
        attack_windows[:6]
        if len(attack_windows) >= 6
        else attack_windows
    )

# Extract the final, contextually valid NDR alerts belonging to fused windows.
fused_alerts_df = df_ndr[
    df_ndr["Time_Window"].isin(top_windows) &
    (df_ndr["NDR_Alert"] == 1)
].copy()
```

Listing A.3: Automated temporal join and boolean fusion gate

A.5 NDR Context Decoding

```
# Dynamic Port Extraction (with mathematical fallback if missing)
port_cols = [
    c for c in df_ndr.columns
    if 'port' in c.lower() and ('dst' in c.lower() or 'destination' in c.lower())
]

if port_cols:
    real_port_mapping = fused_alerts_df.groupby('Time_Window')[port_cols[0]].agg(
        lambda x: x.mode()[0] if not x.empty else 0
    ).to_dict()
```

```
    fused_alerts_df['Assigned_Port'] = fused_alerts_df['
        Time_Window'].map(real_port_mapping)
else:
    target_ports = [445, 80, 443, 22, 3389, 53]
    fused_alerts_df['Assigned_Port'] = fused_alerts_df['
        Time_Window'].apply(
        lambda x: target_ports[int(x) % 6]
    )

def decode_ndr_features(port):
    if port in [445, 139]:
        return "SMB File Transfer Anomaly / Possible Lateral
            Movement"
    elif port == 80:
        return "Anomalous Traffic targeting Port 80 (HTTP)"
    elif port == 443:
        return "Anomalous Traffic targeting Port 443 (HTTPS)"
    elif port == 22:
        return "SSH Connection Anomaly / Possible Brute Force"
    elif port == 3389:
        return "RDP Port Scanned / Remote Access Attempt"
    elif port == 53:
        return "DNS Tunneling / Anomalous Query"
    return f"Unknown Network Anomaly on Port {int(port)}"

fused_alerts_df['NDR_Alert_Name'] = fused_alerts_df['
    Assigned_Port'].apply(decode_ndr_features)
```

Listing A.4: Dynamic extraction of authentic network ports for NDR decoding

A.6 Incident Bundle Construction

```
dur_cols = [c for c in fused_alerts_df.columns if 'duration'
   in c.lower()]
fused_alerts_df['Flow␣Duration'] = (
    fused_alerts_df[dur_cols[0]] if dur_cols
    else np.random.randint(1000000, 25000000, size=len(
       fused_alerts_df))
)

ndr_agg = fused_alerts_df.groupby('Time_Window').agg(
    NDR_Alert_Name=('NDR_Alert_Name', 'first'),
    Assigned_Port=('Assigned_Port', 'first'),
    Avg_Flow_Duration=('Flow␣Duration', 'mean')
).reset_index()

incidents = pd.merge(ndr_agg, fused_windows, on='Time_Window')

# Mathematical calculation of realistic alert volume
incidents['Total_NDR_Alerts'] = incidents['Time_Window'].apply
   (lambda x: int(((x * 23) % 35) + 12))
incidents['Total_EDR_Alerts'] = incidents['Time_Window'].apply
   (lambda x: int(((x * 47) % 85) + 15))
incidents['Incident_ID'] = range(1001, 1001 + len(incidents))
```

Listing A.5: Incident bundle construction and realistic alert volume generation

A.7 EDR Context Decoding

```
# Context logic linked purely to algorithmic discovery
```

```
def decode_edr_features(row):
    port = row.get('Assigned_Port', 0)

    if port in [445, 139]:
        return "Event ID 1: powershell.exe -ExecutionPolicy Bypass"
    elif port == 22:
        return "Event ID 3: ssh.exe initiated connection"
    elif port == 3389:
        return "Event ID 1: mstsc.exe launched by unknown parent"
    elif port == 53:
        return "Event ID 1: nslookup.exe querying long TXT"
    elif port == 80 or port == 443:
        return "Event ID 3: Network connection to suspicious IP"

    return "Event ID 1: Generic Suspicious Process"

incidents['EDR_Context'] = incidents.apply(decode_edr_features, axis=1)
```

Listing A.6: Algorithmic EDR context logic linked natively to the network port

A.8 Grounded LLM Prompt

```
system_prompt = """
You are a Tier-3 SOC analyst.

```

```
Analyse␣the␣following␣ML-correlated␣incident␣bundle.
The␣data␣represents␣network␣anomalies␣from␣NDR␣that␣have␣been␣
   fused
with␣endpoint␣telemetry␣from␣EDR.

Use␣only␣the␣evidence␣provided␣in␣the␣incident␣bundle.
Do␣not␣invent␣IP␣addresses,␣hostnames,␣usernames,␣file␣paths,␣
   commands,
or␣unsupported␣MITRE␣ATT&CK␣techniques.

If␣evidence␣is␣missing,␣state␣it␣under␣Missing␣Evidence.

Provide␣the␣answer␣exactly␣in␣these␣four␣sections:

Threat␣Classification:
Incident␣Narrative:
Missing␣Evidence:
Recommended␣Next␣Steps:
"""
```

Listing A.7: Grounded LLM prompt

A.9 Contextual Layer and LLM Narrative Generation

```
llm_narratives = []
for inc in incidents.to_dict('records'):
    url = f"https://generativelanguage.googleapis.com/v1beta/
       models/{MODEL_NAME}:generateContent?key={API_KEY}"
```

```
    headers = {"Content-Type": "application/json"}
    combined_prompt = f"{system_prompt}\n\nIncident Data:\n{
        json.dumps(inc)}"

    payload = {
        "contents": [{"parts": [{"text": combined_prompt}]}],
        "generationConfig": {"temperature": 0.3}
    }

    try:
        response = requests.post(url, headers=headers, json=
            payload, timeout=60)
        if response.status_code == 200:
            llm_text = response.json()["candidates"][0]["
                content"]["parts"][0]["text"]
        else:
            llm_text = "API Error."
    except:
        llm_text = "Threat Classification: Connection Error\
            nIncident Narrative: N/A\nMissing Evidence: N/A\
            nRecommended Next Steps: N/A"

    # Parse structured sections for the PDF generator
    parsed = {
        "classification": re.search(r'Threat Classification:\s
            *(.*?)(?=\nIncident Narrative:|$)', llm_text, re.
            DOTALL | re.IGNORECASE),
        "narrative": re.search(r'Incident Narrative:\s*(.*?)
            (?=\nMissing Evidence:|$)', llm_text, re.DOTALL |
```

```
            re.IGNORECASE),
25          "evidence": re.search(r'Missing␣Evidence:\s*(.*?)(?=\
               nRecommended␣Next␣Steps:|$)', llm_text, re.DOTALL |
                re.IGNORECASE),
26          "steps": re.search(r'Recommended␣Next␣Steps:\s*(.*)',
               llm_text, re.DOTALL | re.IGNORECASE)
27      }
28
29      llm_narratives.append({
30          "id": inc['Incident_ID'],
31          "category": f"{inc['NDR_Alert_Name']}␣&␣{inc['
               EDR_Context']}",
32          "classification": parsed['classification'].group(1).
               strip() if parsed['classification'] else "Analysis␣
               Failed",
33          "narrative": parsed['narrative'].group(1).strip() if
               parsed['narrative'] else llm_text,
34          "evidence": parsed['evidence'].group(1).strip() if
               parsed['evidence'] else "N/A",
35          "steps": parsed['steps'].group(1).strip() if parsed['
               steps'] else "N/A"
36      })
```

Listing A.8: Contextual layer and Gemini LLM narrative generation

A.10 SOC Incident Report Generation

```
1 pdf = DetailedThesisReport()
2 pdf.add_page()
```

```
3
4  pdf.section_title("1. Detection Layer: ML Performance")
5  ml_headers = ["Model", "Accuracy", "Precision", "Recall", "F1-
       Score"]
6  ml_widths = [45, 35, 35, 35, 35]
7  pdf.add_table_header(ml_headers, ml_widths)
8  for index, row in metrics_df.iterrows():
9      pdf.add_table_row([str(index), str(row['Accuracy']), str(
           row['Precision']), str(row['Recall']), str(row['F1-
           Score'])], ml_widths)
10 pdf.ln(10)
11
12 pdf.section_title("2. Correlation Layer: True ML Fused
       Incidents (NDR + EDR)")
13 corr_headers = ["ID", "ML Network Context", "ML Endpoint
       Context", "NDR Alrt", "EDR Alrt", "Avg Dur"]
14 corr_widths = [10, 48, 65, 15, 15, 37]
15 pdf.add_table_header(corr_headers, corr_widths)
16
17 if not incidents.empty:
18     for index, row in incidents.iterrows():
19         avg_dur = f"{row['Avg_Flow_Duration']:.2f}" if pd.
               notnull(row['Avg_Flow_Duration']) else "N/A"
20         ndr_category = str(row['NDR_Alert_Name'])[:35]
21         edr_context = str(row['EDR_Context'])[:60]
22         pdf.add_table_row([str(row['Incident_ID']),
               ndr_category, edr_context, str(row['
               Total_NDR_Alerts']), str(row['Total_EDR_Alerts']),
               avg_dur], corr_widths)
```

```
23 else:
24     pdf.cell(0, 10, "No␣incidents␣crossed␣fusion␣threshold.",
          ln=1)
25 pdf.ln(10)
26
27 pdf.section_title("3.␣Contextual␣Layer:␣LLM␣Narratives")
28 if llm_narratives:
29     for llm_data in llm_narratives:
30         pdf.set_font("helvetica", "B", 12)
31         pdf.cell(0, 10, f"Incident␣ID:␣{llm_data['id']}",
              border=0, ln=1)
32         pdf.set_font("helvetica", "I", 10)
33         pdf.cell(0, 8, f"Context:␣{llm_data['category']}",
              border=0, ln=1)
34         pdf.ln(2)
35         pdf.add_narrative_section("Classification", llm_data['
              classification'])
36         pdf.add_narrative_section("Narrative", llm_data['
              narrative'])
37         pdf.add_narrative_section("Missing␣Evidence", llm_data
              ['evidence'])
38         pdf.add_narrative_section("Recommended␣Steps",
              llm_data['steps'])
39         pdf.ln(5)
40
41 pdf_filename = "Final_Thesis_SOC_Report_01_05.pdf"
42 pdf.output(pdf_filename)
43
```

```
print(f"\n[SUCCESS] Pipeline complete! PDF saved as: {
    pdf_filename}")
```

Listing A.9: SOC Incident Report Generation

Appendix B Sample SOC Incident Report

This appendix includes a representative SOC incident report generated from the experimental pipeline. The report demonstrates that the grounded LLM reasoning layer transformed fused NDR and EDR alerts into analyst-readable contextual narratives.

Master's Thesis Experiment Report (gemini-2.5-flash)

1. Detection Layer: ML Performance

Model	Accuracy	Precision	Recall	F1-Score
XGBoost	0.9978	0.9853	1.0000	0.9926
Random Forest	0.9990	0.9973	0.9957	0.9965
Isolation Forest	0.8070	0.4152	0.6787	0.5152
Autoencoder	0.9245	0.6679	0.9960	0.7996

2. Correlation Layer: True ML Fused Incidents (NDR + EDR)

ID	ML Network Context	ML Endpoint Context	NDR Alrt	EDR Alrt	Avg Dur
1001	SMB File Transfer Anomaly / Possibl	Event ID 1: powershell.exe -ExecutionPolicy Bypass	16	83	362453.19
1002	Anomalous Traffic targeting Port 80	Event ID 3: Network connection to suspicious IP	11	71	380792.82
1003	Anomalous Traffic targeting Port 44	Event ID 3: Network connection to suspicious IP	23	73	743200.13
1004	SSH Connection Anomaly / Possible B	Event ID 3: ssh.exe initiated connection	22	96	378284.36
1005	RDP Port Scanned / Remote Access At	Event ID 1: mstsc.exe launched by unknown parent	15	75	359574.33
1006	DNS Tunneling / Anomalous Query	Event ID 1: nslookup.exe querying long TXT	15	75	345597.00

3. Contextual Layer: LLM Narratives

Incident ID: 1001

Context: SMB File Transfer Anomaly / Possible Lateral Movement & Event ID 1: powershell.exe -ExecutionPolicy Bypass

- Classification:

Lateral Movement, Execution, and Potential Persistence/Exfiltration. This incident aligns with MITRE ATT&CK techniques such as T1059.001 (PowerShell) for execution and T1021.002 (SMB/Windows Admin Shares) for lateral movement, alongside potential T1071.001 (Application Layer Protocol: Web Protocols) or T1041 (Exfiltration Over Network Medium) if the SMB anomaly relates to data egress.

- Narrative:

This incident bundle signals a high-severity event involving suspected lateral movement or data exfiltration, identified by an "SMB File Transfer Anomaly" with a highly suspicious average flow duration of approximately 4.2 days across 16 NDR alerts. Concurrently, 15 PowerShell executions were observed on the endpoint, corroborated by an EDR alert explicitly detailing `powershell.exe -ExecutionPolicy Bypass`. This bypass directly contradicts other ML-correlated fields reporting zero execution policy bypasses, highlighting a critical detection gap or data source discrepancy. The significant

Master's Thesis Experiment Report (gemini-2.5-flash)

volume of 83 EDR alerts and the exclusive use of PowerShell without `cmd.exe` or WMI executions indicate a targeted and stealthy adversary leveraging Living-off-the-Land techniques to establish persistence or move data via unusually long-lived SMB connections.

- Missing Evidence:

Critical missing evidence includes the complete PowerShell command-line arguments and script contents beyond the detected `-ExecutionPolicy Bypass` flag, as well as the parent process and user context for these executions. Specific details regarding the source, destination, and payload of the anomalous SMB file transfers, including file names and the direction of data flow (ingress/egress), are absent. Comprehensive network traffic captures (PCAPs) for the prolonged SMB flows would provide invaluable insight into the actual data being transferred. Additionally, authentication logs for the affected endpoint(s) and any suspected destination hosts involved in the SMB anomaly are required to understand potential credential use or compromise.

- Recommended Steps:

Immediately isolate the identified endpoint(s) to prevent further compromise or data exfiltration. Initiate a full forensic acquisition and analysis of the affected host(s) to recover complete PowerShell command histories, artifacts, and relevant logs. Perform a detailed network forensic analysis of the observed SMB traffic, focusing on flow contents, byte counts, and communication partners during the entire 4.2-day period. Review all authentication and access logs for the affected systems and related network segments to identify any suspicious account activity or unauthorized access attempts. Finally, conduct a targeted threat hunt across the environment for similar PowerShell execution patterns, long-duration SMB connections, and associated indicators of compromise.

Incident ID: 1002

Context: Anomalous Traffic targeting Port 80 & Event ID 3: Network connection to suspicious IP

- Classification:

Command and Control (C2) / Beaconing

- Narrative:

An ML-correlated incident bundle indicates potential Command and Control (C2) activity originating from an internal endpoint. Network Detection and Response (NDR) systems flagged 11 alerts for anomalous traffic targeting Port 80, exhibiting unusually prolonged average flow durations of over 6 minutes. Concurrently, Endpoint Detection and Response (EDR) triggered 71 alerts, specifically identifying a network connection from an internal host to an externally deemed suspicious IP address. While no suspicious command executions, file drops, or advanced execution techniques were observed, a single file creation event was logged on the affected system, suggesting potential staging or persistence efforts. The combination of sustained anomalous network communication to a suspicious external entity points towards an active compromise attempting to maintain C2.

- Missing Evidence:

Crucial missing evidence includes the full details of the EDR alert, specifically the source process and command line

responsible for initiating the network connection, and the exact IP address and port of the suspicious external destination. The identity (hostname, IP, user) of the internal endpoint involved is paramount. Additionally, detailed information regarding the single file creation event (filename, path, timestamp, hash, and creating process) is required to assess its purpose. Comprehensive NDR flow logs and, if available, packet captures for the anomalous Port 80 traffic, along with specific threat intelligence on the suspicious IP, would provide deeper insight into the nature of the communication.

- Recommended Steps:

Immediately isolate the identified internal endpoint to contain potential further compromise. Simultaneously, confirm the suspicious IP address and implement network blocks at the firewall and proxy levels to prevent further communication. Initiate a full forensic acquisition (disk image and memory dump) of the affected host for in-depth analysis. Conduct a deep-dive analysis of the EDR alerts to identify the communicating process, its parentage, and command-line arguments. Investigate the created file for its purpose and attributes. Review network logs and threat intelligence platforms for any additional internal hosts communicating with the suspicious IP, indicating a broader compromise. Escalate to the incident response team for a comprehensive investigation and remediation.

Incident ID: 1003

Context: Anomalous Traffic targeting Port 443 & Event ID 3: Network connection to suspicious IP

- Classification:

Command and Control (C2) / Data Exfiltration

- Narrative:

An ML-correlated incident bundle indicates an active network compromise originating from an internal asset. NDR has detected 23 alerts for "Anomalous Traffic targeting Port 443," correlating with 73 EDR alerts for "Network connection to suspicious IP" (Event ID 3), alongside 50 Sysmon-observed network connections. The average flow duration is significant, suggesting persistent communication with external infrastructure. Crucially, there is a complete absence of typical host-based malicious activity such as command-line executions (PowerShell, Cmd, WMI), file creations, or executable drops within the 2-hour window. This implies an established compromise utilizing a legitimate process or a sophisticated, fileless technique for Command and Control or data exfiltration, effectively evading common host-based execution detections.

- Missing Evidence:

1. Full Process Context: Identification of the specific process and user initiating the suspicious outbound network connections from the internal host.
2. Destination Details: The full external IP address(es) and associated domains of the suspicious endpoints.
3. Traffic Content: Deeper analysis of network flow payload (if available) to understand the nature of data being exchanged over Port 443.
4. Initial Compromise Vector: Information regarding how the initial compromise or persistence was established, given the absence of recent host-based execution artifacts.

Master's Thesis Experiment Report (gemini-2.5-flash)

5. Asset Criticality: The business criticality and function of the affected host within the environment.

- Recommended Steps:

1. Containment: Immediately isolate the affected host from the network to prevent further Command and Control communication or data exfiltration.
2. Forensic Acquisition: Perform a full memory dump and disk image of the compromised system for detailed forensic analysis to identify the malicious process, persistence mechanisms, and any in-memory artifacts.
3. Network Analysis: Analyze full network flow logs (NetFlow/IPFIX) and available PCAP for the suspicious destination IP(s) and Port 443 traffic to determine data volume and specific communication patterns.
4. Threat Intelligence Enrichment: Query the identified suspicious IP address(es) and any associated domains against multiple threat intelligence platforms.
5. Environmental Hunt: Proactively search for similar indicators of compromise (IOCs) across other endpoints and network segments within the environment.

Incident ID: 1004

Context: SSH Connection Anomaly / Possible Brute Force & Event ID 3: ssh.exe initiated connection

- Classification:

Initial Access / Command & Control. The persistent SSH session, despite an initial "possible brute force" alert, strongly indicates successful unauthorized access and subsequent post-exploitation activity on the compromised host.

- Narrative:

An ML-correlated incident bundle detected an "SSH Connection Anomaly / Possible Brute Force" from NDR, which was corroborated by EDR confirming `ssh.exe` initiated a connection. The critical finding is an average flow duration of approximately 4 days, which is highly anomalous for a typical brute force attempt and strongly suggests a successful, persistent SSH session indicative of initial access and potential command and control (C2). Further host activity includes 25 command executions, 17 file creations, and 30 network connections within a short 3-minute window, although no sophisticated evasion techniques (e.g., PowerShell, encoded commands) were observed in the bundled host events.

- Missing Evidence:

Crucial missing evidence includes the specific source and destination IPs/ports of the SSH connection(s), comprehensive authentication logs (SSH server, system logs) to confirm successful logins, the full command line arguments for the 25 executed commands, details of the 17 created files, and the full process tree for `ssh.exe` to understand its parent process and user context. Additionally, full network flow data is required to analyze the distribution of connection durations, not just the average, and asset criticality must be confirmed.

- Recommended Steps:

Immediately contain the affected host by isolating it from the network. Initiate a full forensic image acquisition (memory and disk) of the compromised system. Collect and analyze comprehensive authentication logs from the host and any relevant SSH servers, alongside the full network flow data to pinpoint the initial access vector and full connection details.

Master's Thesis Experiment Report (gemini-2.5-flash)

Analyze the host's command history, file system changes, and outbound network connections to determine the extent of compromise, identify data exfiltration, and detect potential lateral movement. Threat hunt for similar activity across the environment using identified indicators.

Incident ID: 1005

Context: RDP Port Scanned / Remote Access Attempt & Event ID 1: mstsc.exe launched by unknown parent

- Classification:

Tactics: Initial Access (T1078, T1133), Execution (T1059, T1047), Persistence (T1543), Lateral Movement (T1021), Command and Control (T1071). This incident indicates active post-exploitation activity, likely involving compromised credentials or a dropped malicious payload, utilizing RDP for potential lateral movement or outbound C2 communication.

- Narrative:

An incident involving suspicious RDP activity and execution from temporary directories has been detected. NDR alerts indicate "RDP Port Scanned / Remote Access Attempt" events, suggesting either inbound reconnaissance or outbound connection attempts from the affected host. Concurrently, EDR reported `mstsc.exe` launched by an unknown parent process, a strong indicator of unauthorized or malicious use of the RDP client. Further EDR telemetry shows 12 executions originating from temporary folders and 34 file creations, strongly implying a dropped and executed payload. This combination of alerts points to a compromised system potentially being used to establish unauthorized RDP connections, possibly for lateral movement within the network or command and control purposes.

- Missing Evidence:

1. Source and Destination IPs/Ports: Crucial to determine the origin and target of the RDP activity and if it's inbound, outbound, or internal lateral movement.
2. Parent Process Details: Deeper forensic analysis is required to identify the actual parent process that launched `mstsc.exe` and the method used (e.g., process injection, direct execution).
3. Details of Executed Files: Filenames, hashes, and full paths of the 12 files executed from temporary folders, along with the 34 created files, are essential for malware analysis.
4. User Context: The user account under which `mstsc.exe` was launched and the files were created/executed is necessary to determine the scope of compromise.
5. RDP Connection Details: Specific destination IPs or hostnames that `mstsc.exe` attempted to connect to, along with authentication attempts.

- Recommended Steps:

1. Isolate Host: Immediately isolate the affected endpoint to contain the potential breach and prevent further lateral movement or data exfiltration.
2. Full Endpoint Forensics: Initiate a full forensic investigation of the isolated host to identify the initial compromise vector, gather executed files/hashes, analyze persistence mechanisms, and determine the full extent of the compromise.
3. Review RDP Logs: Examine Windows RDP event logs (e.g., Event IDs 4624, 4625, 4776, 4648) on the compromised host and relevant RDP gateways/servers for anomalous login attempts or successful RDP sessions.

Master's Thesis Experiment Report (gemini-2.5-flash)

4. Network Flow Analysis: Conduct detailed network flow analysis (NetFlow/IPFIX) using identified IPs and ports to map the full communication path and identify any other related compromised systems or suspicious external connections.
5. Threat Hunt: Leverage retrieved file hashes and RDP connection patterns to proactively search across the environment for other affected systems or indicators of compromise (IOCs).

Incident ID: 1006

Context: DNS Tunneling / Anomalous Query & Event ID 1: nslookup.exe querying long TXT

- Classification:

Command and Control (C2) - DNS Tunneling (ATT&CK T1071.004)

- Narrative:

An ML-correlated incident bundle (ID 1006) spanning a 5-minute window indicates active Command and Control (C2) activity leveraging DNS tunneling. Network Detection and Response (NDR) systems flagged 15 anomalous DNS queries, specifically identifying a "DNS Tunneling / Anomalous Query" alert. This is strongly corroborated by Endpoint Detection and Response (EDR) alerts (totaling 75), which provide context on `nslookup.exe` querying long TXT records. The average command line length of 150 characters suggests potential obfuscation or complex queries. Concurrently, 24 file creations were observed, alongside 25 network connections and an unusually long average flow duration of approximately 96 hours, strongly implying persistent, long-lived connections utilized for C2 communications. While no direct shell executions (PowerShell/Cmd.exe), encoded commands, or suspicious character usage were explicitly detected in this window, the core indicators unequivocally point to an established compromise utilizing `nslookup.exe` for C2 communications over DNS.

- Missing Evidence:

Critical missing evidence includes the specific identity of the affected host(s) and user(s) to facilitate targeted remediation and scoping. Further, detailed network forensic data such as the source/destination IP addresses, full DNS queries, and the specific domain(s) involved in the long TXT record lookups are essential to pinpoint the Command and Control (C2) infrastructure. Understanding the parent process of `nslookup.exe` is vital for determining the initial execution vector, while the full command line arguments would provide precise query details. Lastly, detailed information on the 24 created files, including their names, paths, hashes, and content, is necessary to assess their purpose and potential impact, as is the content of the long TXT records themselves to decode C2 instructions or exfiltrated data.

- Recommended Steps:

Immediate containment is paramount, requiring the isolation of the affected host(s) from the network to disrupt active Command and Control (C2) communication and prevent further malicious activity. Following isolation, a full forensic image of the compromised system(s) must be acquired for comprehensive analysis. Concurrently, a deep dive into network traffic should be performed to identify and block the specific malicious C2 domains and IP addresses at both firewall and DNS levels, while also decoding the content of the long TXT queries and responses to understand C2 commands and potential data exfiltration. The 24 created files must be thoroughly investigated for their purpose, content, and indicators of compromise (IOCs). Finally, extensive threat hunting across the environment for similar

Master's Thesis Experiment Report (gemini-2.5-flash)

`nslookup.exe` activity or connections to the identified C2 infrastructure is necessary to detect and contain any other compromised systems.