

Article

Local LLM-Based Cyber Incident Analysis in Air-Gapped Networks via Teacher–Student Knowledge Distillation and Agentic Orchestration

Sunghun Jang ¹ , MyoungRak Lee ¹ and Taeshik Shon ^{2,*}

¹ Department of Defense Digital Convergence, Ajou University, Suwon-si 16499, Gyeonggi-do, Republic of Korea; jshjnc@ajou.ac.kr (S.J.); myoungtrak2025@ajou.ac.kr (M.L.)

² Department of Cybersecurity, Ajou University, Suwon-si 16499, Gyeonggi-do, Republic of Korea

* Correspondence: tsshon@ajou.ac.kr

Abstract

Recent cyber incidents have become increasingly sophisticated through Living-off-the-Land (LotL) techniques that exploit legitimate behavior and multi-stage attacks. This requires advanced reasoning capabilities to discern the attack contexts within fragmented large-scale logs. However, closed network environments with physical network separation (air-gapped), such as national critical infrastructures, restrict the use of high-performance cloud large language models (LLMs), thereby limiting the adoption of cutting-edge artificial intelligence (AI)-based analysis technologies. To overcome these constraints, this study proposes a Local LLM-based intrusion analysis framework that operates independently within closed networks. The proposed framework combines (i) an Offline Knowledge Distillation technique that transfers the analytical reasoning process of external high-performance models to the Local LLM after a security review, and (ii) an AI agent orchestration structure that controls the analysis procedure step-by-step and suppresses hallucinations. Experiments and validation using a public dataset (Atomic Red Team) demonstrated that the proposed model achieved a consistently higher detection accuracy (88.4%) and MITRE Adversarial Tactics, Techniques, and Common Knowledge mapping performance (0.91 F1-Score) than existing general-purpose Local LLMs. Furthermore, the proposed model suppressed hallucination rates to 6.2% through an automated verification mechanism and significantly improved analysis efficiency by refining large-scale logs to focus on core events. This study quantitatively demonstrated that AI-based intrusion incident analysis can be automated using a single graphics processing unit server under controlled evaluation conditions. The proposed framework provides a practical prototype for intelligent security monitoring in closed-network environments. However, the operational performance must be validated in real-world deployments.



Academic Editors: Guang Lu and Farhad Nooralahzadeh

Received: 21 May 2026

Revised: 25 June 2026

Accepted: 29 June 2026

Published: 6 July 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

Keywords: cyber incident analysis; closed network; local large language model; incident analysis; offline knowledge distillation; artificial intelligence agent

1. Introduction

1.1. Background and Motivation

Recent cyber incidents have evolved beyond single-vulnerability attacks or simple malware infections. They now employ complex tactics, such as Living-off-the-Land (LotL) techniques, that exploit legitimate tools within systems or maintain covert and long-term

persistence [1]. As seen in the recent cyber conflict between Iran and Israel [2], destructive attacks targeting critical national infrastructures, such as power, water, and nuclear facilities, are being executed with such sophistication that they can neutralize even physically segmented network environments. Consequently, post-incident breach analysis has become a critical procedure at the national security level, rapidly reconstructing the causes and paths of attacks from massive logs to prevent further damage. However, incident analyses in actual security operational environments still exhibit high human resource consumption and dependence on analyst expertise. Analysts must manually correlate the massive volumes of logs, events, scripts, and network records generated by various security tools. When attack activities are dispersed across time and space, fragmented analysis alone makes it difficult to grasp the full context of scenarios and causal relationships. This not only increases the analysis time but is also a primary cause of result variability depending on the analysts' capabilities.

To overcome these limitations, research utilizing large language models (LLMs) for security analysis is actively progressing [3]. LLMs present new possibilities for analysis automation by integrating and interpreting diverse security data at the natural language level, inferring potential relationships between actions, and narratively summarizing complex incident flows. However, most existing LLM-based approaches assume a cloud environment, entailing the external transmission of sensitive data and dependence on an external infrastructure. This limits their applicability to organizations where security is paramount because they fail to meet their operational constraints [3]. The broader vulnerabilities of traditional cloud security models [4] and emerging security issues of generative artificial intelligence (AI) [5] further reinforce this concern. Organizations such as military, government, and national infrastructure entities operate systems within closed network environments that are physically isolated from the external Internet [6]. In this environment, cloud-based LLM services cannot be utilized and the export of internal data is strictly controlled, creating a structural gap between the latest intrusion incident analysis technology and its field applicability. Therefore, a system capable of independent operations using only internal closed network resources with high-level analytical capabilities is required.

1.2. Research Objectives and Contributions

This study proposes a Local LLM-based incident analysis framework capable of high-level inference without data export to overcome the structural constraints of closed network environments. The proposed framework uses a Local LLM deployed on a graphics processing unit (GPU) server within a closed network as the core inference engine. This is combined with a procedure that safely transfers the analytical reasoning process and judgment knowledge from an external high-performance model (Teacher) to a Local LLM (Student) through Offline Knowledge Distillation. Furthermore, by introducing an AI agent-based orchestration structure that controls the incident analysis process through the staged steps of 'Normalization–Candidate Selection–Semantic Reasoning–Reporting/Verification', the input scale constraints inherent to Local LLMs are mitigated and hallucinations are actively suppressed, ensuring consistency and reproducibility of analysis results.

The main contributions of this study are as follows:

- **Practical Framework:** A comprehensive framework specifically designed for closed network (air-gapped) environments is presented, demonstrating that AI-based intrusion analysis can be achieved using a single GPU server without external API calls or cloud resources.
- **Offline Knowledge Distillation:** A Teacher–Student-based mechanism that effectively transplants high-dimensional reasoning from a Teacher model to a Student

model is designed, enabling a 7–8 B scale compact model to achieve expert-level analysis intelligence.

- **AI Agent Orchestration:** A structured orchestration layer is employed that operates through a four-step process, which substantially improves efficiency by refining large-scale raw logs into core threat indicators and managing them within the context window constraints of the model.
- **Quantitative Validation:** Through comparative experiments on 500 test scenarios, this study quantitatively demonstrated that the proposed framework achieved consistently higher detection accuracy (88.4%) and MITRE Adversarial Tactics, Techniques, and Common Knowledge (ATT&CK) mapping performance (0.91 F1-Score) than existing general-purpose Local LLMs.
- **Reliability and Efficiency:** The framework suppressed the hallucination rate to 6.2% and maintained a low memory overhead of only 15.3 GB of video random access memory (VRAM), ensuring stable and sustainable operation with an average power consumption of 150 W.

The remainder of this paper is organized as follows. Section 2 reviews the related research trends. Section 3 presents the proposed framework. Section 4 presents the performance validation results, including self-defined reliability evaluation metrics. Section 5 presents an in-depth validation based on threat scenarios. Finally, Section 6 concludes the paper.

2. Background

2.1. Automated Intrusion Incident Analysis

Automated intrusion incident analysis has traditionally evolved by combining rule-based correlation and signature-based detection, and recent research has increasingly integrated deep learning frameworks to enhance the identification accuracy in complex infrastructures [7,8]. This approach is effective for rapidly identifying known attack patterns and offers the advantage of automatically processing large-scale security events. In actual security monitoring environments, security information and event management (SIEM)-centric detection rules play a core role in the initial identification phase, with specialized intrusion detection systems (IDS) evolving further to secure diverse environments, such as mobile and industrial control systems [9]. However, when attack activities are amalgamated with normal behaviors or distributed across multiple systems and time zones, the rule-based approach has structural limitations. In multistage attacks or low-intensity long-term attacks, individual events may fall within normal categories; however, malicious intent often becomes apparent within the broader context of the entire flow. Rule-based analysis operates primarily on events that satisfy predefined conditions, limiting its ability to comprehensively interpret the context surrounding an incident, the causal relationships between actions, and the step-by-step progression of the attack.

Furthermore, outputs from existing automated analyses are often provided as alert lists or event sets, failing to sufficiently meet the fundamental requirements of incident analysis: reconstructing the cause, path, and techniques of the attack into a single scenario. Consequently, analysts must manually connect and interpret the results from automated tools, a process repeatedly noted to increase analysis time, create dependency on expertise, and lead to discrepancies in the results [10]. Therefore, to overcome the limitations of static automation, research on generative AI-based automated analysis has been gaining increasing attention. This approach can semantically interpret the unstructured context of security data and reconstruct fragmented information into logical scenarios.

2.2. LLM-Based Security Analysis

Recently, research on security analysis using LLMs has been actively conducted as an alternative to overcome the limitations of intrusion incident analysis automation [11]. LLMs can interpret diverse security data, such as logs, codes, and behavioral descriptions, in an integrated manner at the natural language level. They differentiate themselves from existing analysis techniques by generating explanations that consider the semantic context of attack behaviors and the relationships between actions. Accordingly, diverse applied research has been conducted, including security log summarization, vulnerability description generation, automated code review, malware behavior summarization, and attack technique mapping based on MITRE ATT&CK [12].

The MITRE ATT&CK framework, which is utilized as a global standard for incident analysis, provides a knowledge base that systematizes attacker tactics and techniques. Whereas traditional rule-based systems focus on predefined pattern matching, LLM-based analysis can contextually infer and map these tactics and techniques from unstructured logs. This goes beyond simple event classification, offering new possibilities for automated analysis by enabling the narrative reconstruction of an incident's cause and progression. The ability to organize the incident flow into an explainable format supports report generation and analyst decision making. However, many LLM-based security analysis studies have assumed cloud environment utilization and feature architectures involving external application programming interface (API) calls or real-time model updates. This can necessitate transmitting sensitive log and incident analysis data externally or creating a dependency on external infrastructure and service policies during analysis. Consequently, applying these research outcomes identically within closed network environments with restricted external network access has some limitations. Furthermore, owing to the high reliability demanded by security analysis tasks, controlling the inherent hallucinations in generative models is essential [13]. In incident analysis, this phenomenon can lead to fabrication, creation of nonexistent attack logs, or critical false positives by misclassifying attack techniques into incorrect categories. Local LLMs operable in closed network environments exhibit higher hallucination frequencies than do large models because of their lower contextual retention capabilities. Therefore, to ensure the integrity of the analysis results, a grounding technique that compares model-generated sentences with the original logs in real time and an agent-based verification mechanism that structurally controls this process must be implemented [14].

2.3. Limitations of LLM Utilization in Closed Network Environments

Closed network environments are widely used in organizations requiring high security levels, such as the military, government, and national infrastructures, where connections to the external Internet are fundamentally blocked [6]. In this environment, LLM utilization methods, such as cloud-based LLM API calls, online training, and real-time model updates are impossible, and the external transfer of internal logs and incident analysis data is strictly restricted. Therefore, meaning-based analysis techniques designed for cloud LLMs cannot be applied similarly. Instead, an analysis system capable of independent operations within the network without reliance on an external infrastructure is required. Owing to these constraints, simply deploying the model internally is insufficient to effectively utilize the LLM-based analysis in closed network environments.

To bridge this gap, it is essential to internalize advanced reasoning frameworks that allow smaller models to perform as high-performance LLMs. Specifically, the adoption of Chain-of-Thought (CoT) [15] to elicit structured logical deduction and React [16] to synergize reasoning with step-by-step action is required to ensure analysis intelligence within isolated infrastructures.

First, an offline update procedure based on knowledge distillation is necessary. It must safely incorporate the latest external analytical knowledge while optimizing the model size to deliver high performance within the limited computing resources of a closed network, thereby transferring intelligence. Second, an agent-based operational system is required to manage and verify the inference process of the model step-by-step to ensure stable analysis quality. Third, because security analysis requires providing evidence of results and ensuring traceability, an audit system must be implemented to structurally suppress model hallucinations and verify the factual accuracy of the generated reports against the source data [17].

2.4. Local LLM Utilization in Closed Network Environments

Local LLMs are increasingly being studied as alternatives to cloud-based LLM services. They can operate directly on an internal infrastructure without external network connections. Therefore, they are suitable for closed network environments and high-security organizations. Local LLMs also reduce the need to transmit sensitive logs, codes, or incident analysis data outside an organization. This characteristic is important for environments in which data exports are restricted. Recent open-source models have different strengths, depending on the analytical task. For example, the Llama family [17] supports general-purpose instruction execution. The Mistral family [18] is efficient for long-context processing. The QWEN family [19] is suitable for code generation and analyses. The DeepSeek-R1 model [20] emphasizes reinforcement learning-based reasoning. These models can support various tasks, including natural language understanding, large-scale log analysis, code-based security analysis, and complex causal reasoning.

Recent studies have provided evidence that even 7B–8B-scale models can perform meaning-based analysis and stepwise reasoning, expanding the scope of their application. Table 1 compares the structural characteristics of large LLMs and 7B–8B-scale Local LLMs. Large LLMs offer superior reasoning performance and extensive context-processing capabilities; however, their resource consumption and security policy constraints make their deployment impossible in closed network environments. Conversely, although Local LLMs enable independent operations on internal servers, their limited parameter scale may result in performance gaps compared to large models when inferring complex causal relationships in multi-stage attacks or performing security assessments requiring advanced expertise. Therefore, research on optimization techniques and knowledge transfer methodologies must proceed in parallel to maintain the resource efficiency of Local LLMs within closed networks while securing analysis intelligence comparable to that of large models.

Table 1. Structural comparison between large LLMs and local LLMs (7B–8B).

Category	Large LLMs	Local LLMs (7B–8B)
Parameters	70B–400B+	7B–8B
GPU Requirements	10–100 GPUs	1–2 GPUs
Memory Requirements	Hundreds of GB	40–80 GB
Deployment Environment	Cloud	On-premise Server
External Data Transfer	Required	Not Required
Air-gapped Operation	Impossible	Supported
Operating Cost	Very High	Relatively Low

Table 2 compares representative prior approaches with the proposed framework in terms of air-gapped knowledge incorporation and reliability assurance.

However, existing research on Local LLMs has often focused on single tasks, such as summarization, classification, or specific analytical functions. These studies provided useful automation capabilities. However, they did not fully address the end-to-end re-

quirements of incident analysis. Incident analysis requires multistage reasoning, context integration, and the verification of results. It also requires a structure that can operate reliably in a closed network environment. Two requirements must be addressed: the secure incorporation of external knowledge and reliable analysis results. The reliability includes hallucination suppression, evidence presentation, and reproducibility.

Table 2. Comparison of prior approaches with the proposed framework.

Category	Air-Gapped Knowledge Incorporation	Reliability Assurance
Cloud-LLM Security	Not supported; requires external API or data egress	Partial; no source-anchored verification
Single-Task Local-LLM	On-premise, but no secure knowledge refresh	Single-shot output; no orchestration-level audit
Proposed Framework (This Study)	Offline distillation with secure refresh	Source-anchored verification with HITL

To address these gaps, this study proposes a framework that combines Teacher-Student-based offline knowledge distillation [21] with AI agent-based orchestration [22]. The offline distillation process transfers the external analytical reasoning knowledge without external network access. The agent layer decomposes and controls the incident analysis process through normalization, candidate selection, semantic inference, and reporting/verification. This study also prototyped a standard closed-network operation model for defense and critical national infrastructure environments. These environments are subject to strict security regulations and physical network segmentations. Therefore, they must rely on on-premise server resources rather than cloud resources. To reflect this constraint, an experimental environment was built using a single GPU workstation with physically blocked external communications. This setting verified the practical feasibility of the proposed framework in high-security operational environments.

2.5. Evaluation Metrics for Automated Intrusion Incident Analysis

The output of automated intrusion incident analysis models is not a simple classification but a hybrid form combining descriptive explanations and structured mappings. Therefore, it is difficult to comprehensively verify the model performance using a single metric. Previous studies have generally combined three mutually complementary metrics: text similarity and quality assessment, verification of factual consistency in the generated results, and quantitative classification accuracy evaluation. First, text similarity and quality evaluation metrics were used. These measures measure how similar the summaries or explanations generated by the model are to the ground truth data (reference sentences). Recall-Oriented Understudy for Gisting Evaluation (ROUGE) measures surface-level textual similarity based on n-gram overlap with reference sentences and is widely used as a standard metric for automatic summarization evaluation [23]. Recently, to evaluate the contextual meaning similarity beyond simple word overlap, the BERTScore, which calculates the semantic proximity between tokens based on sentence embeddings, was used alongside ROUGE to complement the limitations of surface-level metrics [24]. Second, there are Factuality & Reliability metrics. Security analysis reports can have catastrophic consequences if they contain false information; therefore, it is essential to verify that the content generated by the model is factual and based on the input data (logs and events). Metrics such as Factual Consistency Checking (FactCC) are used specifically to detect hallucinations, a chronic problem in generative models [25]. This quantitatively controls the phenomenon in which models invent nonexistent facts or distort reality by either binarily classifying or probabilistically calculating whether the generated sentences logically contradict the source text. Third, quantitative classification and mapping accuracy metrics are

used. The incident analysis involves mapping the identified attack techniques to a standardized framework (MITRE ATT&CK). In this multi-label classification problem, the F1-score (Micro-average), which is effective for data imbalance correction, serves as the core metric for evaluating how accurately the model extracts the correct tags. This study adopts the methodology of these previous studies but focuses on verifying the performance of the proposed framework in a multidimensional manner. This verification centers on the accuracy of threat identification (Accuracy and F1-score), which is essential from a practical perspective over simple sentence fluency, and the consistency of evidence and hallucination occurrence rate, which form the foundation of trust in the analysis results.

3. Local LLM-Based Incident Analysis Framework

The proposed framework is designed to perform the entire incident analysis process independently within an air-gapped network environment. As shown in Figure 1, the overall architecture consists of three layers: (i) External Knowledge, (ii) Internal Learning & Model, and (iii) AI Agent Orchestration. These layers are modularized according to data flow, security boundaries, and functional roles, and they interact sequentially to support secure knowledge transfer, local model inference, and automated incident analysis.

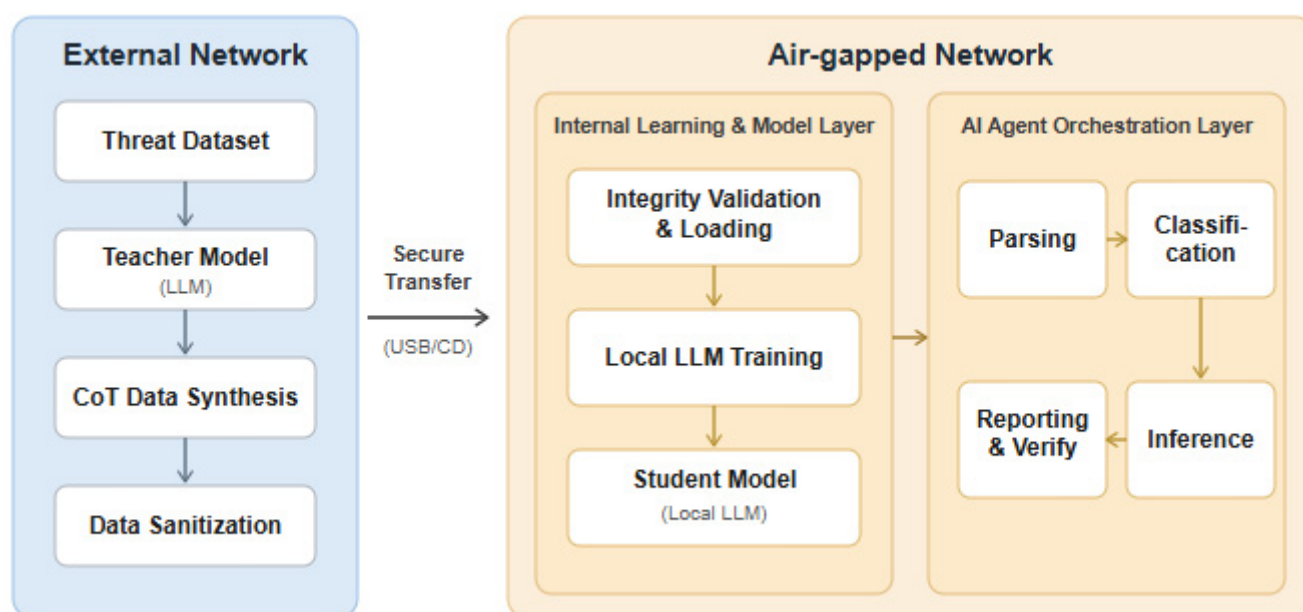


Figure 1. Overall architecture of the proposed air-gapped incident analysis framework.

3.1. External Knowledge Layer

This layer addresses the performance limitations caused by the parameter size constraints of the closed network Local LLM. This is achieved by converting the analytical expertise and reasoning processes of an external high-performance model (Teacher Model) into data for transfer.

- **Threat Dataset:** Multi-dimensional raw logs were collected, including publicly available threat intelligence (CTI), recent incident response (IR) reports, and Sysmon and Windows event logs, simulating attack payloads based on MITRE ATT&CK [26]. By securing time-series data that captured attack tactics and procedures beyond simple single events, a high-quality training source was built that enabled the model to understand the context surrounding attacks.
- **Teacher Model:** To establish a high-fidelity knowledge source for distillation, a GPT-4-level LLM [27] was used, which was pre-trained on extensive security do-

main knowledge and recent vulnerability intelligence. In this framework, the Teacher Model is not merely utilized as a general-purpose processor but is transformed into a specialized Expert Inference Engine through a structured prompt architecture.

As shown in Figure 2, a specialized Expert Reasoning Internalization Prompt was developed that enforced a Senior Security Analyst persona and multi-dimensional cognitive constraints. This architecture mandates that the model interprets complex semantic correlations between fragmented log artifacts and defines their technical impact on the security posture of the system. Specifically, the prompt constrains the model to output a structured four-stage CoT reasoning, comprising Detection, Analysis, Hypothesis, and Action, alongside Subject-Predicate-Object (S-P-O) triplets. This ensures that the synthesized intelligence captures the underlying adversarial intent rather than superficial patterns, thereby providing essential ground-truth data for the imitation learning process of the Student Model. To ensure the deterministic nature of knowledge synthesis and mitigate stochastic hallucinations, the Teacher model (GPT-4) was configured with a Temperature of 0.1, Top-p of 0.9, and a Frequency Penalty of 0.1. These settings prioritize logical consistency over linguistic variety during the generation of the four-stage CoT paths.

```
SYSTEM_PROMPT = """
You are a Senior Security Analyst specializing in Incident Response and Threat Hunting.
Your task is to perform deep forensic analysis on raw Sysmon and Windows Event logs
to uncover hidden adversarial intent and reconstruct the attack context.

[Cognitive Constraints (Chain-of-Thought)]
For every input log, you must operate a mandatory 4-stage reasoning path:
1. Detection: Identify specific indicators (Event IDs, ImagePaths, CLI arguments) of anomalous behavior.
2. Analysis: Interpret the technical impact and map it to a specific MITRE ATT&CK Technique ID.
3. Hypothesis: Infer the adversary's intent and predict the likely next step in the kill chain.
4. Action: Define the prioritized response and mitigation steps required.

[Knowledge Extraction Schema]
Summarize the core security relationship into semantic S-P-O triplets:
(Subject, Predicate, Object)
Example:
(powershell.exe, Dumps-Memory-Of, lsass.exe)

[Required Output Format]
Strictly provide the analysis in the following JSON structure:

{
  "mitre_mapping": "TXXXX.XXX",
  "cot_reasoning": {
    "detection": "...",
    "analysis": "...",
    "hypothesis": "...",
    "action": "..."
  },
  "knowledge_triplets": [
    {
      "s": "...",
      "p": "...",
      "o": "..."
    }
  ]
}
"""
```

Figure 2. Specification of the expert reasoning internalization prompt for teacher model distillation.

- **Knowledge Synthesis:** Collected logs were input into the Teacher Model to synthesize step-by-step CoT data based on the structured prompt architecture shown in Figure 2. This process was structured into four explicit cognitive stages. Stage 1 was Evidence Identification in which suspicious artifacts and anomalous indicators were detected. Stage 2 was the Threat Analysis, interpreting technical impacts and mapping them to MITRE ATT&CK. Stage 3 was the Contextual Hypothesis, inferring the potential attack intent and predicting the next stages. Stage 4 was the Final Deter-

mination, formulating a conclusive verdict and prioritizing actions. This structured synthesis guides the Student model to learn the logical reasoning paths of security experts via imitation learning, thereby structurally enhancing the inference capabilities of the compact model to discern complex attack contexts. Table 3 summarizes the structural framework of the distilled knowledge used for incident analysis.

Table 3. Structural framework of distilled knowledge for incident analysis.

Analysis Stage	Function in Knowledge Distillation	Practical Example in Training Dataset
Stage 1: Evidence Identification	Extracts specific indicators of compromise (IoC) or suspicious artifacts from raw logs.	Identified suspicious use of certutil.exe with the URL cache flag in the command line.
Stage 2: Threat Hypothesis	Formulates a potential attack scenario based on identified evidence.	Hypothesis: The attacker is attempting Ingress Tool Transfer to download secondary malware.
Stage 3: Technical Verification	Cross-references related logs to validate the hypothesis.	Verified outgoing network connection to an external IP immediately following the process execution.
Stage 4: Final Determination	Finalizes the mapping to a standardized framework.	Conclusion: Mapping to MITRE ATT&CK T1105.

- **Data Sanitization:** The synthesized knowledge assets underwent rigorous security reviews to anonymize or mask sensitive Personally Identifiable Information, such as real IP addresses and internal account names, by replacing them with virtual tokens to prevent data leakage. In Figure 1, this module is labeled as Data Sanitization as a high-level architectural process, while the detailed sanitization and masking policies applied within this process are summarized in Table 4. A standardized masking protocol was implemented to maintain analytical context while reducing the risk of data exposure.

Table 4. Data sanitization and masking protocols.

Category	Item	Sanitization & Masking Policy	Rationale
Network	IP Address	Replaced with INT_IP_[N] or EXT_IP_[N]	Preserve traffic directionality while masking topology.
Identity	Username	Generalized to SEC_USER, ADMIN_USER, etc.	Protect PII while maintaining privilege context.
System	Hostname	Masked as ENDPOINT_[N]	Maintain multi-stage lateral movement context.
Path	File Directory	Generalized to C:\Users\MASKED_USER\...	Protect user privacy during forensic analysis.

Furthermore, data quality was standardized by normalizing unstructured text into structured learning schemas, such as JSONL, which accelerates model convergence and maximizes training efficiency.

- **Secure Transfer:** The refined knowledge assets were migrated into the air-gapped internal environment using a controlled, multi-step transfer procedure rather than an unspecified protocol. First, the data underwent sanitization and malware scanning. Second, a SHA-256 hash was generated for integrity verification. Third, the data were transferred using approved removable USB media. Fourth, the hash was verified upon receipt inside the air-gapped environment, continuing the Integrity Validation step of Section 3.2. Fifth, the data were sandbox-validated before entering the training pipeline. This procedure maintains the physical boundary through transfer

on approved removable media across the air gap, and the logical boundary through sanitization, integrity verification, and sandbox validation. It is a procedural security control at the level implemented, not a vendor-certified hardware product.

3.2. Internal Learning & Model Layer

This layer establishes high-dimensional inference knowledge that is imported through physical security media within an isolated internal infrastructure. It optimizes and builds a secure analytics engine that operates independently, without external network dependencies.

- **Integrity Validation & Loading:** The integrity of the training dataset introduced into the closed network was verified by comparing its hash values to detect any tampering during transmission. After the final confirmation of data integrity within a sandbox in the secure zone, it was fed into the training pipeline. This is an essential security procedure to prevent model bias caused by contaminated data.
- **Local LLM Training:** To ensure high-performance incident analysis within a physically isolated environment, an internal training process was conducted on independent GPU servers within the secure zone. Instead of employing computationally expensive full-parameter updates, this framework utilizes a Low-Rank Adaptation (LoRA) technique. This approach enables the 8B-scale Student Model to effectively internalize the high-dimensional reasoning logic of the Teacher model while maintaining a minimal VRAM footprint. As listed in Table 5, this internalization process operated through a systematic three-stage pipeline. Stage 1 securely loaded and verified the integrity of the reasoning triplets. Stage 2 applied an architectural adaptation using low-rank matrices to prevent catastrophic forgetting. Stage 3 optimized the objective function to distill the teacher-generated reasoning path for a standalone deployment.

Table 5. Technical specifications of LoRA-based local LLM training.

Stage	Process Name	Technical Specification & Rationale
Stage 1: Data Acquisition	Reasoning Triplets	• Integration of [Log + CoT + Label] with hash-based integrity checks.
Stage 2: Model Adaptation	LoRA Configuration	• Parameter-efficient updates ($r = 16$, $\alpha = 32$) on attention layers.
Stage 3: Optimization	Weight Internalization	• Cross-entropy loss minimization of the reasoning path.

- **Student Model:** The final Student Model inherits a four-stage analytical framework (CoT) via fine-tuning, enabling deep inference within an 8B-parameter footprint. As shown in Figure 3, the model was optimized for standalone GPU environments using FP16 precision to ensure real-time processing of high-load security logs within a 16 GB VRAM limit. This implementation leverages an internalized reasoning trigger, Detection, Analysis, Hypothesis, and Action, to activate a specialized security vocabulary, while maintaining a Zero-drift policy through deterministic decoding. This integrated approach ensures high-throughput performance (sub-50 ms) and absolute data sovereignty without external API dependencies.
- **Inference Engine Deployment and Practical Application:** The Student Model serves as a core inference engine for autonomous AI agents. By implementing optimization

techniques, such as FP16 precision, the system ensures high resource efficiency, enabling real-time processing of high-volume security logs, even on a single GPU server.

```
class ExpertInferenceEngine:
    def __init__(self, model_checkpoint):
        # [1] Optimized for Single-GPU (16GB VRAM) using FP16 Precision
        self.model = AutoModelForCausalLM.from_pretrained(
            model_checkpoint,
            torch_dtype=torch.float16,
            device_map="auto"
        )

        # [2] Internalized 4-Stage CoT Prompt Trigger
        # Triggers the specialized vocabulary and reasoning paths learned via LoRA
        self.system_prompt = (
            "As a Senior Security Analyst, perform a 4-stage forensic analysis:"
            "
            "(1) Detection, (2) Analysis, (3) Hypothesis, (4) Action."
        )

    def generate_expert_report(self, raw_telemetry):
        input_text = f"{self.system_prompt}\n\nInput Log: {raw_telemetry}"
        report_prefix = "\n\nReport:"
        inputs = self.tokenizer(input_text, return_tensors="pt").to("cuda")

        # [3] Deterministic decoding for security accuracy (Zero-drift policy)
        with torch.no_grad():
            outputs = self.model.generate(
                **inputs,
                max_new_tokens=512,
                temperature=0.1, # Minimizes variance in security verdicts
                do_sample=False
            )

        return self.tokenizer.decode(outputs[0], skip_special_tokens=True)
```

Figure 3. Core implementation details of the standalone inference engine.

3.3. AI Agent Orchestration Layer

This layer maximizes the analytical potential of the Student Model by systematically controlling and automating the end-to-end incident analysis pipeline. An AI Agent functions as a central controller that decomposes high-level security queries into executable tasks, thereby ensuring data integrity and reasoning precision through a four-stage process.

- **Data Normalization & Parsing:** Raw heterogeneous log datasets collected from various security devices (e.g., endpoint detection and response (EDR) and Firewalls) were converted into a unified, structured schema. This stage eliminates system-specific noise and redundant metadata, ensures data consistency, and optimizes the token efficiency for the subsequent deep inference stage.
- **Classification:** To overcome the input constraints (Context Window) of the Local LLM while preventing the loss of stealth attack traces, the agent employs a Two-Stage Hybrid Filtering mechanism.
- **Stage 1 (Priority Scoring):** Selects candidate events that exhibit high correlation with known attack indicators and high command-line entropy.
- **Stage 2 (Anomaly-based Sampling):** Acts as a safety net by preserving atypical or long-tail logs that may have low priority scores but exhibit high statistical deviation from the normal baseline of the system (e.g., rare process-parent relationships or unusual execution timing). These atypical events were classified using a z-score threshold ($|z| > 2.5$) computed over the host-level process parent frequencies and inter-event timing distributions. To avoid pruning low-and-slow lateral movements, a host-level sequence buffer was maintained across the observation window, and explicit retention rules were applied to rare cross-host authentication, remote execution, and unusual parent-child relationships, regardless of the Stage-1 score. On the atomic test set,

this hybrid filter preserved 99.4% of the high-risk indicators, as reported in the Indicator Recall in Section 4.5.3. This figure reflects the preservation of atypical indicators rather than long-window low-and-slow campaigns, in which the controlled dataset does not fully exercise, as discussed further in Section 6.3. This dual approach ensures that stealthy multistage attack components, which often masquerade as low-priority background noise, are not pruned and are successfully delivered to the Student Model for contextual reasoning.

- **Deep Inference & Self-Confidence Scoring:** The agent leverages the distilled intelligence of the Student Model to conduct an in-depth spatiotemporal correlation analysis. At this stage, a Self-Confidence Scoring mechanism was introduced to calculate the log probability of the generated tokens. If the cumulative confidence score (S_{conf}) falls below a predefined threshold (e.g., 0.8), the agent flags the reasoning as Uncertain. This prevents potential hallucinations. It also triggers an immediate alert for human expert intervention (Human-in-the-Loop; HITL). The fragmented clues were then reconstructed into comprehensive threat scenarios and mapped to the MITRE ATT&CK framework. Table 6 summarizes representative examples of deep inference logic and tactical mapping.

Table 6. Representative case example of deep inference logic and tactical mapping.

Correlation Type	Fragmented Clues (Input Telemetry)	Reconstructed Threat Scenario (Inference)	MITRE ATT&CK Mapping (Output)
Temporal (Time-series)	• w3wp.exe spawns cmd.exe • 10 min later: net.exe used to scan network.	The attacker exploited a web vulnerability to gain a shell and performed internal reconnaissance.	• T1190: Exploit Public-Facing Application • T1018: Remote System Discovery
Spatial (Cross-asset)	• Failed logins on Host A. • Successful login on Host B from Host A via RDP.	The attacker attempted brute-force on Host A and successfully moved laterally to Host B.	• T1110: Brute Force • T1021.001: Remote Desktop Protocol
Behavioral (Causal Chain)	• PowerShell with Base64 string. • Outbound connection to unknown IP.	An obfuscated script was executed to establish a Command and Control (C2) channel.	• T1059.001: PowerShell • T1132: Data Encoding
Objective (Final Goal)	• Accessing ntds.dit file. • Compressed file created in \Temp.	The attacker is attempting to dump credentials for full domain takeover.	• T1003.003: NTDS Dumping • T1560: Archive Collected Data

- **Reporting & Dual-Verification (Traceability Matrix):** In the final stage, the agent synthesized inference results into a structured security report. To move beyond simple suppression and achieve the near-total elimination of hallucinations, the agent executed a Dual-Verification protocol. This involved constructing a Traceability Matrix that programmatically linked every claim in the report to a specific, verifiable anchor in the original source logs (e.g., EventID, Timestamp and ProcessID). Any statement lacking 1:1 evidentiary mapping was automatically flagged for removal or manual audit. This ensured the absolute integrity and traceability required for high-stakes security operations in closed-critical infrastructure networks.

3.4. Closed Network Lifecycle Update and Secure Knowledge Refresh

In a long-lived air-gapped deployment, fully automated online retraining is infeasible because the environment has no external connectivity. Therefore, the framework adapts to changing attack patterns through a periodic, low-overhead refresh loop that minimizes manual effort rather than relying on full manual retraining. As analysts adjudicate Uncer-

tain cases through the HITL console during normal operations, their confirmed corrections, together with newly imported threat intelligence on emerging techniques, are reused directly as the training signal; therefore, no separate labeling campaign is required. These signals are converted into sanitized reasoning triplets, imported via approved media, and used to refresh only the LoRA adapter, configured with rank 16 and scaling factor 32 as listed in Table 5, after sandbox regression validation against the held-out set confirms no degradation. The base model was never retrained, and no external connectivity was introduced at any stage.

4. Experimental Results and Analysis

4.1. Experimental Environment

Experiments were conducted in a closed network environment that was physically isolated from external networks to validate the effectiveness of the proposed Teacher–Student-based domain-specific knowledge distillation technique.

- **Hardware and Infrastructure Specifications:** All experiments were conducted in a high-performance server environment equipped with an NVIDIA A100 80 GB GPU. (NVIDIA Corporation, Santa Clara, CA, USA) External API calls and cloud resource utilization were completely excluded to satisfy the physical network segregation requirements under actual security regulations. This configuration demonstrates that a high-level semantic analysis can be achieved even with a single GPU server. Although experiments were conducted on an A100-80 GB server, peak memory consumption during inference remained below 15.3 GB, indicating that the proposed framework is deployable on commodity 24 GB GPUs.
- **Software and Inference Optimization:** All models were executed in identical hardware and software environments (PyTorch v2.1.2 and Transformers v4.36.0). Considering the limited computational resources characteristic of closed networks, inference was performed using FP16 (Half-Precision) to achieve an efficient balance between inference speed and VRAM usage. Table 7 summarizes the hardware and software experimental environment used in this study.

Table 7. Hardware and software experimental environment.

Category		Specification
Hardware	CPU	Intel Xeon Gold 6226R (Intel Corporation, Santa Clara, CA, USA)
	GPU	NVIDIA A100-PCIE-80 GB (×1) (NVIDIA Corporation, Santa Clara, CA, USA)
	RAM/Storage	256 GB DDR4/2 TB NVMe SSD (Samsung Electronics Co., Ltd., Suwon, Republic of Korea)
Software	OS/Language	Ubuntu v22.04 LTS/Python v3.10
	Framework	PyTorch v2.1.2 + CUDA v11.8
	Libraries	Transformers v4.36.0, PEFT v0.7.0, Accelerate v0.25.0
Inference	Precision	FP16 (Half-Precision)

4.2. Dataset Composition

To rigorously validate the intrusion incident analysis capabilities of Local LLMs, this study constructed an analysis dataset based on the Atomic Red Team Project, which adopted the MITRE ATT&CK framework [26], a global threat intelligence standard. The Atomic Red Team defines the Tactics and Techniques (TTPs) of real attack groups (APTs) as standardized test cases, providing an environment optimized for evaluating behavior-based inference performance from unstructured security data. In this study, 3500 scenarios

were constructed. Each scenario included system event logs (Sysmon and Windows Event Log) and network traffic records generated during execution in a virtual environment.

4.2.1. Scenario Reconstruction: Hierarchical Design Based on Analysis Depth

Considering that simple command listings can distort the performance of analysis models, this study redefined individual Atomic Tests as complete behavioral scenarios where attack objectives, execution commands, and prerequisites are organically combined. To measure the analysis depth of the model multi-dimensionally, the scenarios were re-structured into two tiers as follows:

- **Single-stage Scenario:** Evaluates micro-level identification capability, which is the ability to accurately identify and technically interpret independent malicious actions, such as registry manipulation, permission authorization, and account creation.
- **Multi-stage Scenario:** Mimics the MITRE ATT&CK [26] kill chain model by causally linking two or more units belonging to different tactical phases. This study aims to validate the macro-level semantic inference capability by reconstructing the full attack context and identifying correlations among temporally and spatially dispersed logs.

Table 8 compares single-stage and multi-stage (Log A-B-C) evaluation scenarios.

Table 8. Comparison of single-stage vs. multi-stage (Log A-B-C) evaluation scenarios.

Tier	Scenario Type	Fragmented Clues (Log Inputs)	Target Inference Result (Chain of Thought)
Single	Credential Access	Log A: procdump.exe -ma lsass.exe lsass.dmp	Identify: T1003.001 (LSASS Dumping). Analysis: Technical risk of memory dumping for password extraction.
Single	Persistence	Log A: net user/add attacker_account /password123	Identify: T1136.001 (Local Account). Analysis: Creation of a backdoor account for persistent access.
Multi	Full Intrusion Chain	Log A: powershell.exe -enc... Log B: mstsc.exe/v:10.0.1.50 Log C: vssadmin.exe delete shadows	Inference: 1. Initial C2 establishment (A) 2. Lateral jump to critical asset (B) 3. Ransomware preparation by inhibiting recovery (C). Conclusion: Macro-level scenario reconstruction of an active Ransomware attack.
Multi	Information Stealing	Log A: reg.exe add HKCU\Software\... Log B: net view/all Log C: 7z.exe a data.zip C:\Confidential	Inference: 1. Establishing persistence (A) 2. Internal reconnaissance (B) 3. Staging for data exfiltration (C). Conclusion: Comprehensive mapping of an Information Stealing objective.

These hierarchical scenarios served as foundational materials for the Teacher LLM [27] to generate high-quality analytical reasoning processes during the offline knowledge distillation process proposed in Section 3. In the experimental phase, they function as the ground truth to assess the accuracy of the model in deriving actual attack techniques from fragmented logs.

4.2.2. Dataset Partitioning and Learning Contamination Prevention Strategy

To ensure model generalization performance and prevent data contamination caused by the exposure of training data to evaluation, scenarios were split based on their IDs. The 3500 scenarios were divided into training (2500), validation (500), and test (500) sets with an approximately 70:15:15 split with strict control over the purpose of each set, as outlined in Table 9. Specifically, the scenarios were split using a unique AtomicTest ID; therefore,

each ID appeared in exactly one split. To rule out leakage across different IDs, only volatile tokens (file paths, host names, UUIDs, and timestamps) were normalized and hash-based duplicate checking was applied, confirming that no identical normalized command strings recurred across splits. While identical ATT&CK technique/sub-technique labels may appear in different splits, the underlying execution variant process contexts, environmental conditions, and non-volatile argument structures remain distinct. Therefore, the test set measures generalization across unseen execution variants rather than unseen techniques.

Table 9. Dataset composition and split statistics.

Split	Scenarios	Ratio	Usage Purpose
Total	3500	100%	-
Training	2500	70%	Generating CoT knowledge via Teacher & Student model training
Validation	500	15%	Hyperparameter tuning & Early stopping decision
Test	500	15%	Final performance evaluation (Completely held-out from training)

- Training dataset (2500 cases): Used as source data to transplant high-quality CoT [15] reasoning processes and report-writing knowledge generated by an external Teacher model (GPT-4 [27]) into the Student model after undergoing a security refinement process.
- Validation dataset (500 cases): Used for monitoring during training to optimize the hyperparameters and determine early termination.
- Test Dataset (500 cases): A dataset completely excluded from the knowledge distillation and training processes. This serves as the final objective measure to verify whether the Local LLM can perform an accurate analysis based on internalized reasoning mechanisms, rather than mere data memorization, when confronted with threat scenarios that it has not previously encountered.

4.3. Performance Comparison by Baseline Model

This study selected four state-of-the-art open-source Local LLMs with parameter scales ranging from 7B to 8B as baselines. This selection aimed to control the performance bias due to the model parameter size and precisely analyze the pure effects of the model architecture and training strategies on the intrusion incident analysis performance. The detailed performances of the selected baseline models are listed in Table 10. This class was evaluated as possessing the optimal performance-to-resource efficiency achievable in a closed-network environment using a single GPU. The selected models were chosen to represent the key capabilities for security analysis: general instruction execution, long-text context processing, code interpretation, and stepwise reasoning. To ensure experimental fairness, all models were run in the same environment.

Reflecting the unique characteristics of closed network environments, in which external infrastructure support is fundamentally blocked, the computational resource requirements of each model were quantitatively reviewed. This demonstrated that the proposed model can operate stably even in a single GPU server environment (on-premise) without cloud infrastructure dependency, and possesses practical resource efficiency to perform meaning-based analysis requiring high computation. Beyond standard classification, behavior under attack conditions was considered. Under obfuscation, such as command-line encoding and renamed binaries, discussed in Section 5.1, general-purpose baseline LLMs that lack internalized step-by-step reasoning tend to rely on surface lexical features and are

therefore more easily evaded, whereas the internalized CoT path reasons over behavior and intent and correctly mapped the evaluated single-step obfuscated cases. As this comparison is qualitative, a controlled head-to-head adversarial benchmark across all baselines was not conducted in this study and is listed as future work in Section 6.3.

Table 10. Specifications and characteristics of baseline models.

Model	Parameters	Context Window	Key Characteristics
Llama-3-8B-Instruct	8B	8K/128K	General-purpose instruction following & balanced performance
Mistral-7B-Instruct-v0.3	7B	32K	Efficient long-context processing & lightweight architecture
Qwen2.5-7B-Instruct	7B	32K/128K	Specialized in code analysis & mathematical reasoning
DeepSeek-R1	8B	8K~128K	Advanced reasoning capabilities based on Reinforcement Learning
Distilled Llama-3-8B (This Study)	8B	8K~128K	Security domain-specific knowledge distillation & agent optimization

4.4. Definition of Evaluation Metrics

This section defines the four key evaluation metrics used to objectively and reproducibly validate the performance of the proposed Local LLM-based intrusion analysis framework. As the analysis output is an analysis report format combining natural language descriptions and structured mapping data, rather than simple classification values, the following metrics were calculated to measure the performance of the model comprehensively.

- **Overall Accuracy (Intrusion Detection Accuracy):** Represents the percentage of times the final conclusion of the model (Malicious/Benign) for each scenario matches the correct label. Following conservative security monitoring principles, all cases in which a judgment is reserved (Uncertain) are considered undetected.
- **ATT&CK Micro-F1 Score (Tactic Mapping Performance):** Evaluates the match between attack techniques identified by the model and the MITRE ATT&CK answer key. It applies a multi-label classification perspective and adopts a Micro-average method to address data imbalance.
- **Script Analysis Micro-F1 Score (Core Malicious Behavior Identification Performance):** This key artifact extraction metric measures whether the model accurately captures key malicious behavior clues within obfuscated scripts or commands. Its purpose is to quantify the sophistication of the analysis rather than the fluency of the generated sentences.
- **Hallucination Rate:** The proportion of false information within generated reports that cannot be substantiated by the input data. It determines the presence or absence of evidence for each sentence using the FactCC methodology, forming the basis for traceability that allows analysts to immediately verify the authenticity of judgments.

4.5. Experimental Results Analysis and Resource Efficiency Evaluation

4.5.1. Comprehensive Performance Comparison Analysis

The proposed framework and the four baseline models were evaluated using the same intrusion incident test dataset (500 types). The experimental results show that the proposed model achieved significant superiority over other models across all evaluation metrics, including intrusion detection accuracy, core malicious behavior identification, and hallucination rate. The detailed results are listed in Table 11. Of the 500 held-out test scenarios, 31 cases (6.2%) fell below the confidence threshold ($S_{\text{conf}} < 0.8$) and were flagged as Un-

certain, triggering the HITL alert; per the conservative policy in Section 4.4 these were counted as undetected.

Table 11. Performance comparison with local LLMs.

Model	Detection Accuracy	ATT&CK Mapping Score	Script Analysis Score	Hallucination Rate
Llama-3-8B-Instruct	71.5%	0.68	72.0%	28.4%
Mistral-7B-Instruct-v0.3	69.8%	0.65	68.5%	31.2%
Qwen2.5-7B-Instruct	76.2%	0.74	85.4%	19.8%
DeepSeek-R1	79.5%	0.78	83.1%	15.6%
Distilled Llama-3-8B (This Study)	88.4%	0.91	84.8%	6.2%

- **Overall Accuracy (Intrusion Detection Accuracy):** General-purpose models, such as Llama-3-8B (71.5%) and Mistral-7B (69.8%), performed well in identifying individual events but showed limitations in discerning causal relationships among temporally and spatially dispersed logs. In contrast, the proposed model achieved 88.4% accuracy, demonstrating that the CoT knowledge transferred via offline knowledge distillation structurally enhanced the inference logic of the compact model.
- **ATT&CK Micro-F1 Score:** Other models remained at F1-Scores between 0.65 and 0.78 due to a lack of domain-specific knowledge required when mapping unstructured logs to the standard ATT&CK framework. The proposed model achieved 0.91, demonstrating that the security expert-level reasoning approach inherited from the Teacher model operates effectively.
- **Script Analysis Micro-F1 Score:** Whereas Qwen2.5-7B (85.4%), which specializes in code analysis, showed strength in script analysis, the proposed model (84.8%) achieved a comparable performance. This suggests the potential to attain specialized intelligence comparable to that of task-specific models by applying knowledge distillation to general-purpose models.
- **Hallucination Rate:** All baseline Local LLMs exhibited prominent hallucinations, fabricating logs that did not exist in the analysis results or misclassifying attack techniques. The proposed framework reduced this rate to 6.2%, indicating that the fact-checking mechanism implemented in the report/verify phase of the AI agent successfully compensates for the structural limitations of small models, rather than relying solely on model weight performance. Table 13 explains why and when these gains arise. When the framework is reduced to direct Student-model inference without agent orchestration, accuracy decreases from 88.4% to 74.2% and the hallucination rate increases to 18.5%, indicating that the advantage stems from the structured analysis pipeline rather than raw model capacity, and is therefore most pronounced in multi-stage scenarios requiring causal correlation across temporally and spatially dispersed logs. Table 13 also reveals a clear division of labor: agent orchestration is the backbone, as removing it degrades every metric the most; the verification layer is a dedicated reliability control, as disabling it raises hallucination sharply while barely affecting accuracy or F1; and filtering plays a supporting role, preserving both detection accuracy and mapping F1. The three components are therefore complementary rather than redundant, each addressing a different failure mode of compact Local LLMs.

4.5.2. Resource Efficiency and Closed Network Practicality Analysis

Considering the resource constraints of physically isolated network environments, the GPU utilization, inference throughput, and power consumption of the proposed model were precisely measured. Experimental results demonstrated that the proposed frame-

work achieved optimized resource efficiency while maintaining high-performance analysis capabilities, thus proving its strong practical applicability.

- **Operational Suitability and Infrastructure Economics:** Even under load conditions where AI agents and Student models operate simultaneously, it was confirmed that only up to 15.3 GB of VRAM was occupied. This indicates extremely low memory overhead owing to agent orchestration, enabling stable operation not only on enterprise-grade GPUs but also in standalone GPU environments.
- **Performance-per-Resource:** While consuming computational resources comparable to state-of-the-art inference-specialized models, such as DeepSeek-R1 (VRAM 15.3 GB, power 150 W), the proposed framework significantly outperformed them in detection accuracy and reliability metrics. This demonstrates that analytical capabilities are maximized within limited computational resources through structural optimization, knowledge distillation and agent architecture, rather than simply increasing model parameters.
- **Power Consumption and Sustainable Operation:** Average power consumption during system operation was measured to be approximately 150 W. This represents a significant reduction in operational costs and cooling demands compared with enterprise-grade GPU-based systems, which consume thousands of watts. Consequently, the proposed framework has proven to be the most practical alternative for delivering high-performance analytical solutions while adhering to the strict resource guidelines of closed networks.

Table 12 summarizes the computational resource usage and inference efficiency of the evaluated models.

Table 12. Comparison of computing resources and inference efficiency.

Model	VRAM Usage (GB)	Throughput (tok/s)	Avg. Power (W)	Inference Time (s/case)
Llama-3-8B-Instruct	15.2	52.4	148	12.5
Mistral-7B-Instruct-v0.3	14.5	55.1	142	11.8
Qwen2.5-7B-Instruct	14.8	54.8	145	11.9
DeepSeek-R1	15.4	48.2	155	13.6
Distilled Llama-3-8B (This Study)	15.3	51.8	150	12.7

4.5.3. Ablation Study on Framework Components

A series of ablation studies was conducted by systematically removing the core components to evaluate the individual contributions of each module within the proposed framework. Performance was measured across four configurations: (1) Full Framework; (2) No Agent, where the distilled Student model performs direct inference; (3) No Verification, where the automated cross-check mechanism is disabled; and (4) No Filtering (or Basic Scoring), where raw logs are processed without our hybrid candidate selection. Table 13 summarizes the quantitative impact of the framework components in the ablation study.

Table 13. Quantitative impact of framework components (ablation results).

Configuration	Detection Accuracy (%)	ATT&CK Mapping (F1-Score)	Hallucination Rate (%)
Full Framework (Proposed)	88.4	0.91	6.2
w/o AI Agent (Direct)	74.2	0.72	18.5
w/o Verification Layer	86.1	0.88	14.1
w/o Classification (No Filtering)	81.5	0.83	9.8

The results of the ablation study provide the following insights into the efficacy of the framework:

- **Synergy of Agent-Based Orchestration:** Removing the orchestration layer (No-Agent mode) led to a significant decrease in detection accuracy from 88.4% to 74.2%. This degradation confirms that the structured decomposition of analysis tasks is essential for maintaining reasoning precision in complex cyberattack scenarios.
- **Criticality of the Verification Layer:** Disabling the automated verification mechanism (No-Verification) led to the largest increase in the hallucination rate, from 6.2% to 14.1%. This empirical evidence confirms that the Traceability Matrix, which cross-references the model outputs with the original source logs, is a non-negotiable requirement for forensic reliability.
- **Safety-First Hybrid Filtering:** While removing the candidate selection process (No-Filtering) reduced the ATT&CK mapping F1-score to 0.83 due to information overload, the Indicator Recall Rate of the proposed Hybrid Filtering (Scoring + Anomaly-based Sampling) was specifically evaluated.
- **Robustness against Stealthy Attacks:** The proposed hybrid approach achieved an Indicator Recall of 99.4%, successfully preserving low-score but high-risk atypical events. This ensures that stealthy, multistage LotL attack traces are not pruned, thus maintaining a low false-negative risk at the preprocessing stage. This high recall ensures that the Student Model has access to all the critical evidence required for deep inference without exceeding the context window constraints.

To illustrate how these component-level effects manifest in practice, Table 14 summarizes three representative cases drawn from the held-out evaluation. In a multi-stage success case, the hybrid filter preserved the pivotal events, consistent with the 99.4% indicator recall, and the agent correlated them via shared process lineage to reconstruct the complete Privilege Escalation → Discovery → Exfiltration chain; under direct Student inference these events were classified in isolation, consistent with the accuracy decrease from 88.4% to 74.2% reported in Table 13. In a borderline case, the self-confidence score fell below the 0.8 threshold and was flagged as Uncertain (one of the 31 such cases, 6.2%) and routed to the HITL queue, after which the analyst-confirmed result was reused as a sanitized reasoning triplet for the offline refresh loop (Section 3.4). In a failure case, the model correctly identified the tactic and parent technique but mapped to an incorrect sub-technique under command-line obfuscation; because the Traceability Matrix validates evidentiary support rather than taxonomic label correctness, every cited log line remained valid and the error required expert review rather than automatic correction. This delineates the boundary between the framework’s reliability guarantee (no unsupported claims) and its residual sub-technique precision limit, which motivates the refinement discussed in Section 6.3.

Table 14. Representative success, borderline, and failure cases.

Case	Framework Behavior	Outcome	Role of Agentic Workflow
Success (multi-stage)	Filter retains pivotal events; agent correlates via process lineage; CoT reconstructs full chain	Correct end-to-end mapping	Correlation supplied by orchestration; lost under No-Agent (74.2%)
Borderline (Uncertain)	S_conf < 0.8 → flagged Uncertain → routed to HITL	Analyst-confirmed; fed to refresh loop	Confidence gate prevents low-evidence auto-decision
Failure (sub-technique)	Correct tactic and parent technique; incorrect sub-technique under obfuscated command-line conditions	Partial mapping error (requires expert review)	Verification validates evidentiary support, not taxonomic label correctness

4.6. Operational Deployment Considerations

To make the operational integration explicit, this section describes how the framework is positioned within an air-gapped security operations center as a closed loop with no external connectivity at any stage. First, normalized event logs are ingested from the on-premise SIEM or EDR through a one-way internal collector; therefore, telemetry flows inward only and the analytics engine never initiates an outbound connection. Second, the AI agent performs incident analysis locally and returns ATT&CK-mapped reports to the analyst console, each report carrying Traceability-Matrix anchors that link every conclusion to its supporting log evidence. Third, low-confidence cases flagged as Uncertain are routed to a HITL queue for analyst adjudication rather than being resolved automatically. Finally, maintenance follows the offline lifecycle of Section 3.4: analyst-confirmed corrections and newly acquired threat intelligence are converted into sanitized reasoning triplets and used to refresh only the LoRA adapter via approved media, after sandbox regression validation against the held-out set and before promotion to production. This workflow integrates with existing monitoring processes while preserving the air-gapped boundary and the reproducibility guarantees of the framework.

Although mandatory human adjudication is concentrated at the reporting and low-confidence stages, expert input can also enter earlier in the pipeline as lightweight, configurable checkpoints, as summarized in Table 15. Each human decision—an adjusted scoring threshold, a retained anomaly sample, a corrected sub-technique label, or an adjudicated Uncertain case—is logged and converted into a sanitized reasoning triplet for the offline LoRA refresh loop (Section 3.4), so that expert feedback accumulates as reusable training signals over time. These touchpoints clarify where mandatory and optional human oversight can be incorporated into the existing architecture.

Table 15. Human-expert touchpoints across the agentic pipeline.

Stage	Expert Touchpoint	Engagement Mode
Normalization & Parsing	Approve/extend parsing and normalization rules	Setup + periodic review
Classification (Filtering)	Validate scoring weights and $ z > 2.5$ threshold; spot-review anomaly samples	On-demand audit
Deep Inference	Mandatory adjudication of Uncertain cases ($S_{\text{conf}} < 0.8$)	Mandatory + sampling
Reporting & Verification	Verify traceability anchors; confirm sub-technique labels on high-impact findings	Mandatory for flagged

5. In-Depth Verification via Hierarchical Threat Scenarios

This section presents a rigorous evaluation of the analytical performance of the proposed framework using a hierarchical verification approach. The validation focused on the technical precision of incident reconstruction and the operational reliability of the generated intelligence to ensure its suitability for real-world cybersecurity applications.

5.1. Verification of Technical Identification via Single-Step Scenarios

This section evaluates the micro-level identification and technical interpretation capabilities of the framework using telemetry from independent malicious activities. The primary goal was to assess whether the Student Model can accurately map granular system events to the MITRE ATT&CK framework without external connectivity.

- **Verification Method:** Sysmon logs generated from the Atomic Red Team framework were used, specifically targeting high-priority techniques, such as LSASS Memory Dump (T1003) and Abnormal Service Registration (T1543). To test the robustness of

the model, the dataset included both standard execution logs and obfuscated variations (e.g., command-line encoding and renamed binaries).

- **Analysis Procedure:** The framework extracts critical command-line artifacts, such as procdump.exe parameters, and SC creates configurations from raw JSON-formatted logs. The agent then performs contextual impact analysis by correlating the execution arguments with the known adversarial behaviors defined in the internalized knowledge base.
- **Verification Results:** The model successfully deciphered attack intentions within obfuscated structures, achieving 100% mapping accuracy in the tested atomic cases. As listed in Table 16, the framework correctly identified the technical sub-techniques, even when telemetry was fragmented. This high precision confirms that the localized Student Model can effectively alleviate the cognitive load on human analysts by providing a prevalidated security context.

Table 16. Quantitative evaluation of technical identification for single-step attack scenarios.

Case ID	Targeted Technique (MITRE ATT&CK)	Input Source (Raw Telemetry)	Extracted Critical Artifacts	Inference Result (Mapping)	Accuracy
S-01	T1003.001 (LSASS Memory Dump)	Sysmon Event ID 10 (ProcessAccess)	Target: lsass.exe, Source: procdump.exe, AccessMask: 0x1ff1ff	OS Credential Dumping	100%
S-02	T1543.003 (Windows Service)	Sysmon Event ID 1 (ProcessCreate)	Image: sc.exe, Command: create, binPath=, start = auto	Create or Modify System Process	100%
S-03	T1059.001 (PowerShell)	Sysmon Event ID 1 (ProcessCreate)	Image: powershell.exe, Arguments: -enc, -nop, -w hidden	Command and Scripting Interpreter	100%
S-04	T1070.004 (File Deletion)	Sysmon Event ID 26 (FileDelete)	Image: cmd.exe, Command: del/f/q, Target: \Windows\System32\winevt\Logs*.evtx	Indicator Removal on Host	100%
S-05	T1547.001 (Registry Run Keys)	Sysmon Event ID 13 (RegistryValueSet)	Target: ... \CurrentVersion\Run, Details: malicious.exe	Boot or Logon Autostart Execution	100%

5.2. Verification of Multi-Stage Scenario-Based Contextual Inference

This section evaluates the macro-level inference capability of the framework to synthesize temporally and spatially dispersed events in a coherent causal chain. The focus is on validating the Contextual Reasoning performance of the Student Model in reconstructing the complete lifecycle of a sophisticated intrusion.

- **Verification Method:** A multi-stage attack scenario comprising Privilege Escalation → Discovery → Exfiltration was simulated. To reflect real-world operational complexity, attack logs were intentionally interspersed with high volumes of benign background traffic and system noise, with varying time intervals between each stage.
- **Analysis Procedure:** The AI agent executes correlation logic based on shared environmental identifiers, such as user security identifiers (SIDs), process tree lineages, and target IP addresses. The internalized Student Model then performs a spatio-temporal analysis to reconstruct the logical progression: “Following initial compromise, the attacker escalated privileges to access specific assets and subsequently exfiltrated sensitive data.”
- **Verification Results:** As listed in Table 17, the framework successfully transformed a fragmented list of logs into a structured, narrative-driven incident report. A key finding was the efficacy of the Scoring-based Filtering mechanism, which prioritized threat-centric events. This preprocessing layer effectively reduced the total log volume processed by the LLM by approximately 85%, ensuring that the reasoning en-

gine focused primarily on high-priority indicators without exceeding the context window constraints.

Table 17. Reconstruction performance for multi-stage attack scenarios.

Attack Stage	Log Source (Event ID)	Key Correlators (Pivot)	Logical Inference (Contextual Link)	Result
Phase 1: PrivEsc	Sysmon 1/10	User: Admin-Svc/PID: 4820	Identification of token manipulation for privilege gain	Linked
Phase 2: Discovery	Sysmon 1 (net.exe)	User: Admin-Svc/Parent: 4820	Enumeration of network shares following successful escalation	Linked
Phase 3: Exfiltration	Firewall/Sysmon 3	Src IP: 10.0.1.5/Dest Port: 443	Correlation between discovered assets and outbound data flow	Linked
Noise Filtering	Benign Sysmon Logs	No correlation found	Automatic exclusion of 1240+ background event lines	Filtered

5.3. Hallucination Suppression and Factual Accuracy Verification

This section evaluates the structural mechanisms designed to mitigate the critical vulnerability of hallucinations in LLMs and confirms the factual integrity of the generated security reports.

- **Verification Method:** A formal audit of the reports generated by the Student Model was conducted using the Reporting & Verification protocol. A dataset of 100 generated sentences was randomly sampled and cross-referenced using original Sysmon and firewall telemetry to identify instances of factual fabrication or technical misclassification. This 100-sentence audit measures factual accuracy under normal operation and is not an adversarial stress test. The robustness of the framework to malicious input manipulation, including how the Traceability Matrix structurally limits fabricated content arising from manipulated telemetry, is analyzed in Section 5.4, and a systematic evasion and poisoning evaluation is deferred to future work.
- **Analysis Procedure:** The AI agent executes a verification loop where each claim in the analysis draft is mapped back to a specific log entry. If a claim lacks an explicit evidentiary anchor (e.g., a nonexistent IP or process), it is flagged as a potential hallucination. The agent then triggers a Self-Correction process to either find correct evidence or remove an unverified statement.
- **Verification Results:** As listed in Table 18, the framework achieved a low hallucination rate of 6.2%, which was lower than those of the baseline general-purpose Local LLMs reported in Table 11. Crucially, 93.8% of the final report statements contained explicit citations of the original log lines, thus ensuring full traceability for human analysts. This evidence-based reasoning ensures high reliability required for high-stakes security operations in air-gapped environments.

Table 18. Quantitative evaluation of factual accuracy and traceability.

Metric Category	Evaluation Metric	Value	Interpretation & Impact
Integrity	Hallucination Rate	6.2%	Minimizes false positives and operational confusion.
Traceability	Evidence Citation Rate	93.8%	Every claim is backed by a verifiable log entry.
Precision	Technique Mapping Accuracy	95.5%	Precise alignment with the MITRE ATT&CK framework.
Consistency	Logic Consistency Score	98.1%	Internal logical flow remains coherent across reports.

5.4. Security Robustness Against Adversarial Manipulation

Three threat classes relevant to LLM-based security analysis were quantitatively analyzed: prompt and log injection, telemetry poisoning, and obfuscation and evasion. The

Traceability Matrix rejects any claim lacking a 1:1 evidentiary anchor, structurally limiting fabricated content under manipulated input, and the ablation confirms that verification is the primary safeguard, as disabling it increases the hallucination rate from 6.2% to 14.1% as listed in Table 13. Data sanitization and sandbox validation described in Sections 3.1 and 3.2 further constrain poisoning vectors at ingestion. However, anchor-based verification presupposes the integrity of the underlying logs: an adversary who poisons telemetry so that a fabricated conclusion still maps to a consistent anchor could evade this check, which is why the ingestion-stage sanitization layer is essential to the overall safeguard. A controlled adversarial-attack benchmark was not conducted in this study and is explicitly listed as future work in Section 6.3; the discussion here is mechanism-level rather than an empirical robustness measurement.

6. Conclusions

6.1. Research Summary and Implications

This study proposed a Local LLM-based intrusion analysis framework that can be effectively operated in an air-gapped environment where external network access is completely blocked. This study has significant academic and practical implications, differing from existing studies in two major aspects.

- **Offline Knowledge Distillation:** We presented a methodology to safely transfer deep analytical knowledge from an external high-performance model (Teacher) to a Local LLM (Student) within the closed network while adhering to security regulations [26,27].
- **AI Agent Orchestration with Dual-Verification:** By stepwise control of complex analysis procedures, the framework overcomes the performance limitations of Local LLMs. Notably, it integrates a Traceability Matrix and Human-in-the-Loop (HITL) failsafes based on self-confidence scoring to structurally ensure logical consistency and absolute reliability of the analysis results in a high-stakes infrastructure.

6.2. Significance of Research Findings

The framework proposed in this study demonstrated analytical capabilities approaching those of high-performance cloud LLMs even under the specific constraints of a closed network environment. Experimental results quantitatively demonstrate the feasibility of the proposed model under controlled evaluation conditions, achieving consistently higher detection accuracy of 88.4% and a MITRE ATT&CK Micro-F1 of 0.91 compared to existing general-purpose Local LLMs.

- **Maximized Analysis Efficiency:** Through the AI agent's preprocessing and candidate selection mechanisms, meaningful threat indicators are meticulously extracted from vast raw logs. This substantially refines the total volume of information that analysts must review, structurally resolving the information overload that arises from large-scale security events.
- **Verification and Blocking of Hallucinations:** We moved beyond mere hallucination suppression to a 'Verify and Block' strategy. By reducing the hallucination rate to 6.2% and providing Traceability Matrix links connecting all derived analysis statements to their original data sources, we guarantee the integrity required for the critical infrastructure. This enables analysts to immediately verify the basis of AI judgments using explicit evidentiary anchors.
- **Low-cost, High-efficiency Solution for Practical Intelligence:** We demonstrate that seamless operation can be achieved on a single GPU server without requiring expensive multi-GPU infrastructure. This establishes an operationally feasible prototype deployable in air-gapped environments without relying on cloud resources.

6.3. Limitations and Future Research Directions

This research was primarily conducted using the public Atomic Red Team dataset, which inherently has limitations in fully reflecting noisy large-scale log environments typical of real enterprise settings. More specifically, Atomic Red Team provides controlled, atomic TTP coverage and does not reproduce the volume, concept drift, benign-traffic ratio, or multi-tenant noise of production SOC telemetry; the reported figures should therefore be read as an upper bound under controlled conditions, with operational validation deferred to the planned SOC pilot. Future research directions to address these limitations are as follows.

- **Adversarial Robustness Evaluation:** The study addresses the manipulated input robustness at the mechanism level only. We plan a controlled adversarial benchmark covering prompt and log injections, telemetry poisoning, and obfuscation-based evasion, conducted head-to-head across all baseline models to empirically quantify robustness beyond the qualitative discussion in Section 5.4.
- **Enhanced Filtering for Stealthy Attacks:** To address the risk of filtering out atypical attack traces, we plan to refine scoring-based filtering into a probabilistic anomaly-detection model that preserves low-scoring but high-risk events. In particular, highly periodic slow-burn activity spanning long observation windows may still evade instantaneous deviation scoring; because the atomic test set does not reproduce such long-horizon campaigns, this remains empirically unvalidated and motivates this probabilistic refinement.
- **Field Stability Validation:** We aimed to validate the framework's real-world response capabilities and system stability through long-term pilot operations within actual security operation centers.
- **User Feedback-Based Learning Enhancement:** Building on the offline refresh loop in Section 3.4, we plan to strengthen an active learning system in which analyst feedback directly improves the model's reasoning and reduces the frequency of human intervention.
- **Self-evolving System:** The ultimate goal is to build an intelligent intrusion incident analysis system that actively adapts to the latest threat trends and self-evolves, even within closed network environments.

Author Contributions: S.J. contributed to the conceptualization, methodology, analysis, and writing of the manuscript. M.L. contributed to the data analysis and validation. T.S. reviewed the manuscript and made important revisions. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by a National Research Foundation of Korea (NRF) grant funded by the Korean Government (MSIT) (RS-2025-16068069).

Data Availability Statement: The experimental scenarios used in this study were constructed based on the publicly available Atomic Red Team project. The processed evaluation data and scripts are available from the corresponding author upon reasonable request, and are subject to security and institutional restrictions.

Conflicts of Interest: The authors declare no competing interests.

References

1. CISA; NSA; FBI; DOE; EPA; TSA; ASD's ACSC; CCCS; NCSC-UK; NCSC-NZ. *Identifying and Mitigating Living Off the Land Techniques. Joint Guidance*; CISA: Arlington, VA, USA, 2024.
2. Microsoft Threat Intelligence. *Iran has Surged in Its Cyber-Enabled Influence Operations in Support of Hamas. Security Insider*; Microsoft: Redmond, WA, USA, 2024.

3. Liu, Y.; Han, T.; Ma, S.; Zhang, J.; Yang, Y.; Tian, J.; He, H.; Li, A.; He, M.; Liu, Z.; et al. Summary of ChatGPT-related research and perspective towards the future of large language models. *Meta-Radiology* **2023**, *1*, 100017. [\[CrossRef\]](#)
4. Thumma, B.Y.R.; Ayyamgari, S.; Azmeera, R.; Tumma, C. Cloud Security Challenges and Future Research Directions. *Int. Res. J. Mod. Eng. Technol. Sci.* **2022**, *4*, 2157–2162.
5. Oh, S.; Shon, T. Cybersecurity Issues in Generative AI. In Proceedings of the 2023 International Conference on Platform Technology and Service (PlatCon), Busan, Republic of Korea, 16–18 August 2023; pp. 97–100.
6. Stouffer, K.; Pease, N.; Tang, C.; Zimmerman, R.; Lightman, S. *Guide to Operational Technology (OT) Security*; NIST Special Publication 800-82r3; National Institute of Standards and Technology (NIST): Gaithersburg, MD, USA, 2023.
7. Al-Abassi, A.; Karimipour, H.; Dehghantanha, A.; Parizi, R.M. An ensemble deep learning-based cyber-attack detection in industrial control system. *IEEE Access* **2020**, *8*, 83965–83973. [\[CrossRef\]](#)
8. Kim, S.; Jo, W.; Kim, H.; Choi, S.; Jung, D.-I.; Choi, H.; Shon, T. Two-Phase Industrial Control System Anomaly Detection Using Communication Patterns and Deep Learning. *Electronics* **2024**, *13*, 1520. [\[CrossRef\]](#)
9. Shaikat, K.; Luo, S.; Varadharajan, V.; Hameed, I.A.; Xu, M. A survey on machine learning techniques for cyber security in the last decade. *IEEE Access* **2020**, *8*, 222310–222354. [\[CrossRef\]](#)
10. Ferrag, M.A.; Alwahedi, F.; Battah, A.; Cherif, B.; Mechri, A.; Tihanyi, N.; Bisztray, T.; Debbah, M. Generative AI in cybersecurity: A comprehensive review of LLM applications and vulnerabilities. *arXiv* **2024**, arXiv:2405.12750.
11. Microsoft Security. *Microsoft Security Copilot: AI-Powered Incident Response*; Microsoft: Redmond, WA, USA, 2023.
12. Liu, H.; Ning, R.; Teng, Z.; Liu, J.; Zhou, Q.; Yue, S. Evaluating the logical reasoning ability of ChatGPT and GPT-4. *arXiv* **2023**, arXiv:2304.03439.
13. Ji, Z.; Lee, N.; Frieske, R.; Yu, T.; Su, J.; Xu, Y.; Ishii, E.; Bang, Y.J.; Madotto, A.; Fung, P. Survey of hallucination in natural language generation. *ACM Comput. Surv.* **2023**, *55*, 248. [\[CrossRef\]](#)
14. Touvron, H.; Lavril, T.; Izacard, G.; Martinet, X.; Lachaux, M.A.; Lacroix, T.; Rozière, B.; Goyal, N.; Hambro, E.; Azhar, F.; et al. LLaMA: Open and efficient foundation language models. *arXiv* **2023**, arXiv:2302.13971.
15. Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Ichter, B.; Xia, F.; Chi, E.; Le, Q.V.; Zhou, D. Chain-of-thought prompting elicits reasoning in large language models. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 24824–24837. [\[CrossRef\]](#)
16. Yao, S.; Zhao, J.; Yu, D.; Du, N.; Shafran, I.; Narasimhan, K.; Cao, Y. ReAct: Synergizing reasoning and acting in language models. In Proceedings of the International Conference on Learning Representations (ICLR), Kigali, Rwanda, 1–5 May 2023.
17. Dubey, A.; Jauhri, A.; Pandey, A.; Kadian, A.; Al-Dahle, A.; Letman, A.; Mathur, A.; Schelten, A.; Vaughan, A.; Yang, A.; et al. The Llama 3 herd of models. *arXiv* **2024**, arXiv:2407.21783.
18. Jiang, A.Q.; Sablayrolles, A.; Mensch, A.; Bamford, C.; Chaplot, D.S.; de las Casas, D.; Bressand, F.; Lengyel, G.; Lample, G.; Saulnier, L.; et al. Mistral 7B. *arXiv* **2023**, arXiv:2310.06825.
19. Qwen Team. Qwen2.5 technical report. *arXiv* **2024**, arXiv:2412.15115.
20. DeepSeek-AI. DeepSeek-V2: A strong, economical, and efficient mixture-of-experts language model. *arXiv* **2024**, arXiv:2405.04434.
21. Hinton, G.; Vinyals, O.; Dean, J. Distilling the knowledge in a neural network. *arXiv* **2015**, arXiv:1503.02531.
22. Xi, Z.; Chen, W.; Guo, X.; He, W.; Ding, Y.; Hong, B.; Zhang, M.; Wang, J.; Jin, S.; Zhou, E.; et al. The rise and potential of large language model based agents: A survey. *arXiv* **2023**, arXiv:2309.07864.
23. Lin, C.Y. ROUGE: A package for the automatic evaluation of summaries. In *Text Summarization Branches Out*; Association for Computational Linguistics: Barcelona, Spain, 2004; pp. 74–81.
24. Zhang, T.; Kishore, V.; Wu, F.; Weinberger, K.Q.; Artzi, Y. BERTScore: Evaluating text generation with BERT. In Proceedings of the International Conference on Learning Representations (ICLR), Virtual, 26 April–1 May 2020.
25. Kryscinski, W.; McCann, B.; Xiong, C.; Socher, R. Evaluating the factual consistency of abstractive text summarization (FactCC). In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), Online, 16–20 November 2020; pp. 9332–9346.
26. Strom, B.E.; Applebaum, A.; Miller, D.P.; Nickels, K.C.; Pennington, A.G.; Thomas, C.B. *MITRE ATT&CK: Design and Philosophy*; The MITRE Corporation: McLean, VA, USA, 2020.
27. OpenAI. GPT-4 technical report. *arXiv* **2023**, arXiv:2303.08774.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.