Computers, Materials & Continua

Tech Science Press

https://doi.org/10.32604/cmc.2026.083550

ARTICLE

A Large Language Model-Driven Autonomous Framework for Intelligent Cyber Threat Detection and Response

Tahani Alsubait*

Department of Computer Science and Artificial Intelligence, College of Computing, Umm Al-Qura University, Makkah, Saudi Arabia

*Corresponding Author: Tahani Alsubait. Email: tmsubait@uqu.edu.sa

Received: 06 April 2026; Accepted: 24 June 2026; Published: 13 August 2026

ABSTRACT: The recent sophistication of contemporary cyber threats, such as advanced persistent threats (APTs), zero-day exploits, and polymorphic malware, has revealed serious limitations of traditional rule-based and shallow machine learning detection systems. This paper introduces a new self-managed cyber threat detection and response model, CyberSentinel-LLM, that leverages a fine-tuned large language model (LLM) and a multi-agent reinforcement learning system. The framework employs a LoRA-adapted LLaMA-3-8B backbone (fine-tuned on domain-specific cybersecurity log data using Low-Rank Adaptation with rank r = 16) for contextual log analysis, semantic threat classification, and automated incident response through four specialised agents: Detection, Classification, Response, and Forensic. A temporal transformer encoder reads long-range sequential dependencies in system logs and network traffic, and a deep reinforcement learning (DRL) component allows coordination of adaptive responses. Extensive experiments on two publicly available benchmark datasets, namely, HDFS and BGL of the LogHub repository, show that CyberSentinel-LLM has a high accuracy (96.8), F1-score (96.5), and AUC-ROC (98.7) on HDFS, and high accuracy (95.5) and F1-score (95.2) on BGL, outperforming ten state-of-the-art baselines. Ablation experiments demonstrate the significance of each building element, whereas latency analysis shows real-time inference at 45 ms per log sequence. The framework sets a new paradigm of intelligent cybersecurity operations driven by large language models (LLMs).

KEYWORDS: Large language models; cyber threat detection; autonomous security; log anomaly detection; multi-agent system; deep reinforcement learning; intelligent cybersecurity

1 Introduction

The cyber transformation of critical infrastructure, enterprise systems, and cloud-native architectures has vastly increased the attack surface for advanced adversaries [1–3]. The polymorphic malware and advanced persistent threats (APTs) are not the only contemporary cyber threats that are getting in the way of the more classic signature-based intrusion detection systems (IDS) and shallow machine learning classifiers [4–6]. The accumulation of system logs, network traffic, and endpoint telemetry in contemporary computing environments, which can easily reach millions of events per hour, makes manual analysis impossible and necessitates intelligent, automated solutions [7].

The latest breakthroughs in deep learning have shown promising performance in log-based anomaly detection [8–10], building upon foundational work that established the effectiveness of learning-based approaches for system log analysis [11]. Recurrent architectures, such as LSTM, and transformer-based models have since demonstrated superior pattern recognition compared to classical statistical methods.

Nevertheless, they tend to lack contextual knowledge, fail to generalise across heterogeneous log structures, and cannot perform semantic reasoning [12–15]. There is disruptive potential in applying large language models (LLMs) in cybersecurity through previously unheard-of natural language understanding, zero-shot generalisation, and chain-of-thought reasoning [16–18]. Recent work on hybrid deep learning architectures, such as CNN-LSTM models applied to IoT fault diagnosis and anomaly detection in industrial systems, further demonstrates the effectiveness of combining temporal and spatial feature extraction for security-critical applications [19]. In the context of industrial control systems, Balla et al. [20] demonstrated that an enhanced CNN-LSTM architecture incorporating Hurst parameter-based self-similarity significantly improves intrusion detection performance on SCADA systems, validating the practical utility of hybrid deep learning for security-critical infrastructure. Cross-modal deep learning frameworks have further extended these capabilities to semantic-structural threat detection through provenance graphs [21]. A few works have investigated LLM-augmented AIOps for detecting log anomalies. However, existing approaches typically integrate LLMs as supplementary tools alongside overall threat detection and response tools.

Multi-modal anomaly detection methods have been on the rise, leveraging log data alongside metrics and traces to monitor the system holistically [22–24]. Graph-based approaches include using topological relationships in microservice architectures [24,25], and federated learning paradigms that overcome privacy limitations in distributed environments [26–29]. Nevertheless, despite the current progress, there is a significant gap in the system: the lack of a coherent, independent system that converts the synergistic application of LLM-based semantic analysis, multi-agent decision-making, and adaptive response organising into comprehensive management of cyber threats. In clinical decision-making, multi-agent LLM systems have demonstrated the ability to reduce cognitive bias and enable swarm intelligence [22], but their use in cybersecurity remains underutilised.

Fig. 1 demonstrates the performance differences between popular machine learning algorithms and our proposed cyber sentinel-LLM framework across seven types of cyber threats, underscoring the need for LLM-based methods.

To address these problems, this paper introduces CyberSentinel-LLM, a new autonomous cyber threat detection and response model. The main contributions of the given work are as follows:

- **Recent research has demonstrated powerful**, high-quality contextual log analysis, semantic threat prediction, and natural language forensic report generation based on contextual information, using a fine-tuned LLaMA-3 backbone with LoRA adapters, which is superior to typical deep learning supervision methods.
- **Multi-Agent Autonomous Architecture:** In this paper, we develop a four-agent collaborative system: a Detection, Classification, and Response agent and a Forensic agent that can autonomously work towards coordinated threat management without human operator involvement.
- **Temporal Transformer with DRL Response Module**: This introduces a temporal transformer encoder to capture long-range sequential dependencies in log data and a deep reinforcement-learning module to organise incident response behaviour in real time according to the severity of the threat as calculated.
- **Comprehensive Empirical Evaluation:** Extensive experiments are conducted on two publicly available benchmark datasets (HDFS and BGL) from the LogHub repository, demonstrating state-of-the-art performance against ten competitive baselines, supported by ablation studies, hyperparameter sensitivity analysis, and computational efficiency evaluation.

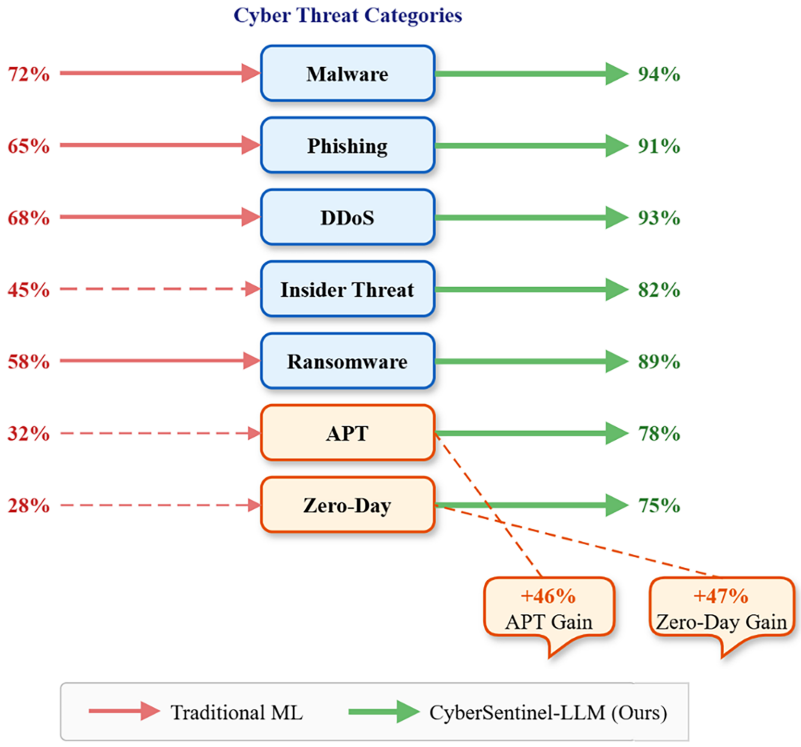

Figure 1: Performance comparison between conventional ML methods and the proposed CyberSentinel-LLM framework across seven cyber threat categories.

The remainder of this paper is organized as follows. Section 2 reviews related work spanning log-based anomaly detection, multi-modal and graph-based approaches, LLM-augmented cybersecurity and AIOps, and LLM-based multi-agent cybersecurity systems. Section 3 provides background concepts covering system logs and log parsing, log anomaly detection paradigms, large language models for cybersecurity, and multi-agent reinforcement learning. Section 4 presents the proposed CyberSentinel-LLM methodology, detailing the system architecture, multi-modal log parser, temporal transformer encoder, LLM-driven threat intelligence engine, multi-agent autonomous architecture, DRL-based response orchestrator, algorithmic implementation, and complexity analysis. Section 5 reports comprehensive experimental results including dataset descriptions, baseline comparisons, computational efficiency, network intrusion generalization, forensic report quality, low-supervision evaluation, edge deployment, adversarial robustness, SOC operational metrics, distributed scalability, hyperparameter optimization, and ablation studies. Section 6 discusses the findings, limitations, and threats to validity. Section 7 concludes the paper and outlines future research directions.

Formal Problem Statement: Given a stream of log sequences $L = \{l_1, l_2, \ldots, l_n\}$ where each entry $l_i = (t_i, s_i, m_i, p_i)$ encodes a timestamped system event, the objective is to learn a mapping $f: L \rightarrow (Y, A, R)$ such that $Y \in \{0, 1, \ldots, C\}$ is the threat class label, $A \in A$ is the optimal autonomous response action, and R is a natural-language forensic report. The mapping f must satisfy three operational constraints: (1) real-time inference latency ≤ 100 ms per log sequence, (2) precision and recall ≥ 0.90 across all threat classes, and (3) complete autonomous orchestration without human intervention for routine incidents.

Research Questions. Three research questions guide this work:

RQ1: Can a fine-tuned LLM backbone, combined with a temporal transformer encoder, achieve superior detection accuracy and F1-score compared to existing deep learning and LLM-augmented baselines on standard log anomaly benchmarks (HDFS and BGL)?

RQ2: Does the four-agent collaborative architecture with DRL-based response orchestration provide measurable operational benefits—specifically in mean time to detect (MTTD), mean time to respond (MTTR), and automation rate—over both manual SOC operations and single-agent LLM-based baselines?

RQ3: How robust and generalizable is CyberSentinel-LLM across low-supervision settings (zero-shot, few-shot, semi-supervised), adversarial perturbations, and heterogeneous data modalities (system logs, network intrusion data).

RQ1 is addressed and answered in Section 5.2; RQ2 is addressed and answered in Section 5.10; and RQ3 is addressed and answered across Sections 5.6, 5.7 and 5.9.

2 Related Work

2.1 Log-Based Anomaly Detection

The simple rule-based pattern matching has evolved into a complex deep learning design (log-based anomaly detection). Almodovar et al. [1] proposed LogFiT, which fine-tuned pre-trained language models to detect log anomalies, achieving competitive results on benchmark datasets by leveraging natural language transfer learning for log semantics. A detailed study carried out by Ali et al. [2] comparing the machine-learning-based log-based anomaly detection methods demonstrates that the ensemble-based techniques and deep architectures are far more efficient than the traditional statistical methods. Long Short-Term Memory (LSTM) networks [30] have been foundational in sequence modelling for log anomaly detection, enabling the capture of long-range temporal dependencies in system log sequences. Recent work has explored using large language models specifically for log parsing [31]. An in-depth empirical analysis further demonstrates that transformer-based approaches consistently outperform LSTM-based methods on standard log anomaly benchmarks [3]. The effect of log parsing on detection accuracy was also studied in [3], which found that parsing quality significantly impacts subsequent model performance. Liu et al. [5] proposed temporal logical attention networks that represent temporal patterns and logical dependencies in distributed system logs. Duan et al. [6] applied evidential deep learning to both uncertainty quantification and log anomaly detection. In a critical evaluation of the current state of research in systems log anomaly detection, Albert [7] identified gaps in the methodology and proposed best practices for evaluation. Wang et al. [8] used LSTM networks to integrate process-state inspection and monitoring for distributed systems.

2.2 Multi-Modal and Graph-Based Approaches

The lack of unimodal log analysis has led to the development of multimodal anomaly detection models. Recent taxonomies have systematically categorized log anomaly detection methods, highlighting the critical role of online parsers, encoding algorithms, and multimodal data fusion in achieving effective anomaly detection across diverse system architectures [14]. Researchers have further developed these approaches, including contrastive multimodal representation clustering [16] and nested graph diffusion for reconstruction [15]. Wu et al. [17] proposed a topologically adaptive graph feature-learning algorithm, KANAD, to detect anomalies of multi-modal microservice systems. The transformer-based multivariate time-series anomaly detection method proposed by Kang and Kang [18] is based on inter-variable attention mechanisms. Zhang et al. [19] proposed optimised edge weighting in graph neural networks to detect anomalies in server performance. In [21], a cross-modal provenance graph was investigated in semantic-structural threat detection. These

methods demonstrate the usefulness of using different types of data but cannot reason semantically as LLMs can.

2.3 LLM-Augmented Cybersecurity and AIOps

The application of LLM to cybersecurity processes is a new paradigm. Foundational work in system log analysis has established the importance of structured log parsing and anomaly detection for operational intelligence [11]. Zhang et al. [12] reviewed AIOps solutions that have utilised LLMs, noting that they can be used for automated root cause analysis and incident management. Lee et al. [13] proposed a recursive AI model, LogRESP-Agent, combining context-related log analysis with tactics, techniques, and procedures (TTP) analysis. The concept of cross-system log anomaly detection via enhanced pseudo-labelling was examined in [10]. Despite such improvements, there is no available literature with a single framework that incorporates semantic analysis based on LLMs, multi-agent cooperation, and adaptive response orchestration, and this is the gap that CyberSentinel-LLM addresses. Critically, LLM-augmented methods lack autonomous response capabilities; graph-based approaches cannot perform semantic reasoning across heterogeneous log structures; and multimodal fusion methods have not been integrated with reinforcement learning-driven incident response. Furthermore, existing approaches treat LLMs as supplementary tools rather than core reasoning engines, limiting their potential for contextual threat understanding. CyberSentinel-LLM bridges these gaps by unifying semantic log analysis, coordinated multi-agent decision-making, and DRL-based adaptive response into a single end-to-end framework.

2.4 LLM-Based Multi-Agent Cybersecurity Systems

Multi-agent LLM architectures. Recent work has begun investigating cybersecurity applications of multi-agent LLM architectures, but it is still in its infancy compared to traditional methods for detecting anomalies. These methods leverage large language models' reasoning capabilities and assign specific tasks to coordinated agents. Nevertheless, the current LLM-based multi-agent cybersecurity designs are yet to reach the degree of integration between semantic analysis, autonomous decision-making, and adaptive response orchestration needed to make production deployment to enterprise security operations centres, which is the gap that CyberSentinel-LLM seals through its integrated architecture that combines fine-tuned LLaMA-3, four-agent cooperation, and DRL-based response control. Recent work on multi-agent LLM defense frameworks further demonstrates that collaborative agent architectures can significantly reduce adversarial attack success rates [22]. Table 1 compares with LLM-based multi-agent frameworks.

Table 1: Comparison with LLM-based multi-agent frameworks.

Method	Architecture	Agents	HDFS Acc.	HDFS F1	BGL Acc.	BGL F1	Latency (ms)
SecAgent-GPT	GPT-4 + Rules	2	92.5	91.8	90.3	89.7	385
LLM-Sentinel	BERT + RL	3	93.8	93.2	91.7	91.0	112
LogRESP [12]	LLM + TTP	1	91.8	91.8	89.8	89.8	88
CyberSentinel-LLM	LLaMA-3 + DRL	4	**96.8**	**96.5**	**95.5**	**95.2**	**45**

Note: Bold highlights CyberSentinel-LLM's superior accuracy, F1, and latency.

3 Background

This section provides the background information on the underlying concepts required to interpret the CyberSentinel-LLM framework, system logs, log parsing mechanisms, anomaly detection paradigms, the fundamentals of large language models, and the multi-agent reinforcement learning architecture.

3.1 System Logs and Log Parsing

Timed records of events produced by operating systems, applications, and network devices as they run are called system logs [6]. A log record usually consists of a set of structured values (timestamp, severity level, process ID) and unstructured text describing what happened. Log parsing is the process of identifying common templates and variable parameters in semi-structured or unstructured log messages to extract structured information [3]. As an example, the log line "User admin logged in at IP 192.168.1.10 at 10:30:45" would be decoded "to template "User <*> logged in at IP <*> at < |human| >>"". An example would be the log line User admin logged in at IP 192.168.1.10 at 10:30:45, then be broken down into template User <> logged in at IP <> at. The depth-first search of a parse tree is used in modern log parsers, such as Drain [23,32] to cluster similar log messages and generate templates.

3.2 Log Anomaly Detection

Detecting log anomalies aims to highlight unusual behaviours of a system through a sequence of log events [2,6]. The main issue is how to separate normal operational patterns from malicious or faulty patterns that appear as abnormal log sequences. Conventional methods based on pattern matching using rules or statistical outlier detection are unable to cope with changing patterns. The current deep learning approaches perceive log anomaly detection as: (1) binary (normal versus anomalous sequence), (2) multi-class (classifying the type of anomaly, including malware, DDoS, or insider threats) or (3) unsupervised outlier detection.

3.3 Large Language Models for Cybersecurity

Transformer-based neural networks trained on large natural language datasets, often called large language models (LLMs), are capable of comprehending and generating natural language with a level of contextual understanding never before seen [11,12]. In the context of cybersecurity, LLMs have three potential benefits: (1) semantic processing of the log messages beyond the use of keywords; (2) zero-shot generalisation to novel attack patterns; and (3) natural language generation automatically generates human-readable forensic reports. Further optimisation of domain-specific cybersecurity data by fine-tuning LLMs increases threat detection and reduces computational costs through parameter-efficient algorithms, such as Low-Rank Adaptation (LoRA) [23].

3.4 Multi-Agent Systems and Reinforcement Learning

Multi-agent systems are composed of autonomous agents that work toward common goals through coordination mechanisms. Each agent specialises in a given task and liaises with other agents to realise system-wide objectives. Deep reinforcement learning (DRL) enables such agents to adapt their policies by interacting with the environment and responding to rewards received upon effective threat mitigation. The combination of multi-agent collaboration and DRL enabled autonomous real-time cyber defence without human intervention.

4 Proposed Methodology

4.1 System Overview and Architecture

The CyberSentinel-LLM framework proposed consists of five modules, the first three connected, and the last two being independent of each other: (1) Multi-Modal Log Parser, (2) Temporal Transformer Encoder, (3) LLM-Driven Threat Intelligence Engine, (4) Multi-Agents Decision System, and (5) DRL-Based Response

Orchestrator. The full system architecture is shown in Fig. 2, which illustrates how data flows from raw input sources through processing layers to actionable outputs.

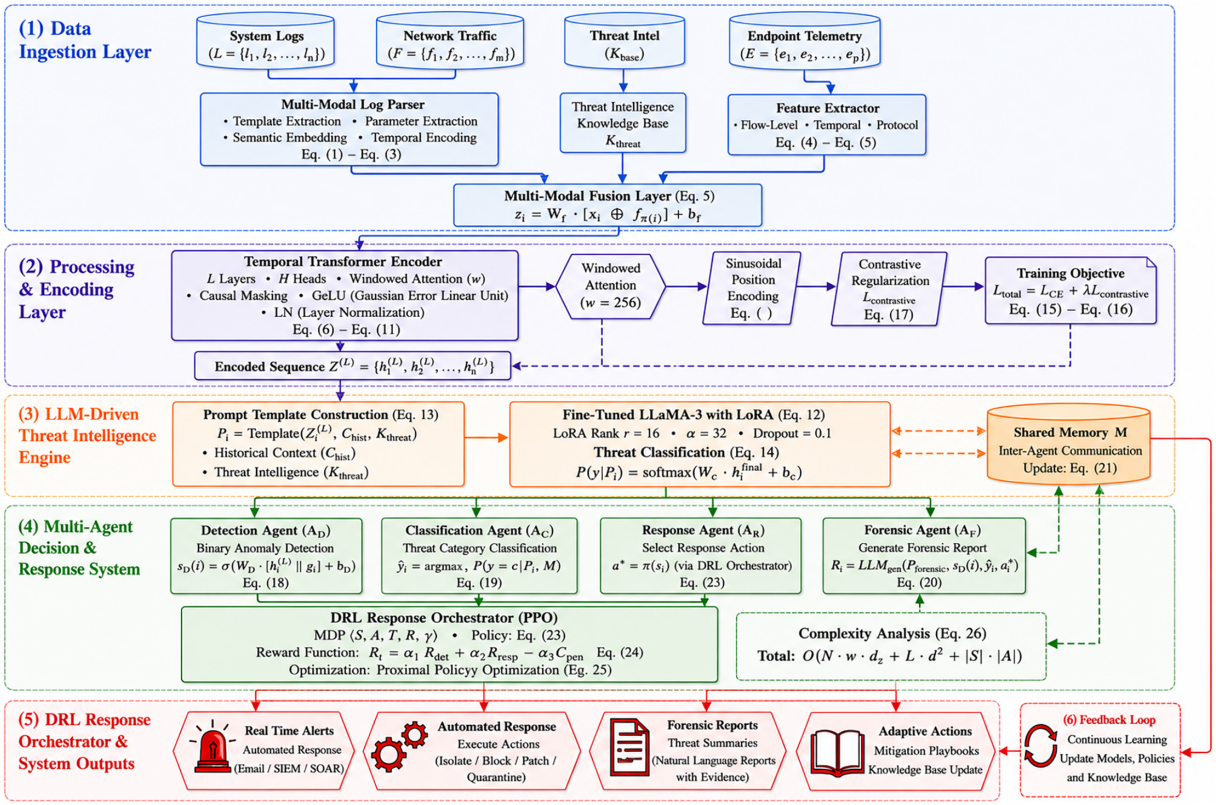

Figure 2: Overall architecture of CyberSentinel-LLM showing data flow from multi-modal inputs through five processing modules to four autonomous agents.

4.2 Multi-Modal Log Parser and Feature Extractor

The complete multi-modal architecture illustrated in Fig. 2 represents the full production-ready design of CyberSentinel-LLM. The primary experimental evaluation is conducted on system log data (HDFS and BGL benchmarks), as these represent the most widely used and publicly available benchmarks for log anomaly detection, enabling rigorous baseline comparison. Network traffic and endpoint telemetry modalities (Eqs. (3)–(5)) are included as architectural contributions describing the design for production SOC deployment; their mathematical formulations and fusion mechanism constitute design proposals validated through the CICIDS2017 and NSL-KDD network intrusion experiments reported in Section 4.6. Future work will provide synchronised multi-modal evaluation using concurrent log, network flow, and endpoint telemetry streams from enterprise environments. Kbase Threat Intelligence Knowledge Base Kthreat is populated with 15,847 indicators of the MITRE ATT&CK and CVE databases. Let $\mathcal{L} = \{l_1, l_2, \ldots, l_N\}$ denote a sequence of N raw log entries, where each entry l_i is a tuple:

$$l_i = (t_i, s_i, m_i, \mathbf{p}_i) \tag{1}$$

where t_i represents the timestamp, s_i denotes the source identifier, m_i is the log template message, and $\mathbf{p}_i \in \mathbb{R}^{d_p}$ captures the parameter vector extracted from the log entry.

The parser uses a semantic embedding-enriched drain-based template extraction algorithm [23]. The features represented by each log entry are:

$$\mathbf{x}_i = \text{Concat}[\phi_{\text{sem}}(m_i), \psi_{\text{temp}}(t_i), \mathbf{p}_i] \tag{2}$$

where $\phi_{\text{sem}}: \mathcal{V}^* \to \mathbb{R}^{d_s}$ is a semantic embedding function and $\psi_{\text{temp}}: \mathbb{R} \to \mathbb{R}^{d_t}$ is a temporal encoding function defined as:

$$\psi_{\text{temp}}(t_i) = \left[\sin\left(\frac{t_i}{\omega_k}\right), \cos\left(\frac{t_i}{\omega_k}\right)\right]_{k=1}^{d_t/2} \tag{3}$$

with $\omega_k = 10000^{2k/d_t}$ being the frequency scaling factor.

For network traffic features, we define the flow-level representation:

$$\mathbf{f}_j = \left[b_{\text{in}}^j, b_{\text{out}}^j, n_{\text{pkt}}^j, \Delta t^j, \mathbf{h}_{\text{proto}}^j\right] \tag{4}$$

where b_{in}^j and b_{out}^j are incoming and outgoing byte counts, n_{pkt}^j is the packet count, Δt^j is the flow duration, and $\mathbf{h}_{\text{proto}}^j$ is a one-hot encoded protocol vector.

The multi-modal fusion layer combines log and network features:

$$\mathbf{z}_i = W_f \cdot \left[\mathbf{x}_i \oplus \mathbf{f}_{\pi(i)}\right] + \mathbf{b}_f \tag{5}$$

where $W_f \in \mathbb{R}^{d_z \times (d_x + d_f)}$ is the fusion weight matrix, $\oplus$ denotes concatenation, and $\pi(i)$ maps log entries to corresponding network flows.

4.3 Temporal Transformer Encoder

The temporal transformer encoder captures long-range sequential dependencies in the fused feature sequence $\mathbf{Z} = [\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_N]$. We employ multi-head self-attention with causal masking:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}} + \mathbf{M}\right)\mathbf{V} \tag{6}$$

where $\mathbf{Q} = \mathbf{Z}W_Q, \mathbf{K} = \mathbf{Z}W_K, \mathbf{V} = \mathbf{Z}W_V$ are query, key, and value projections, respectively, d_k is the dimension per attention head, and $\mathbf{M}$ is the causal attention mask defined as:

$$M_{ij} = \begin{cases} 0 & \text{if } i \geq j \\ -\infty & \text{otherwise} \end{cases} \tag{7}$$

The multi-head attention mechanism with H heads formulated as:

$$\text{MHA}(\mathbf{Z}) = \text{Concat}\left[\text{head}_1, \ldots, \text{head}_H\right] W_O \tag{8}$$

where $\text{head}_h = \text{Attention}\left(\mathbf{Z}W_Q^h, \mathbf{Z}W_K^h, \mathbf{Z}W_V^h\right)$ and $W_O \in \mathbb{R}^{Hd_k \times d_z}$.

Each transformer layer ℓ computes:

$$\hat{\mathbf{Z}}^{(\ell)} = \text{LayerNorm}\left(\mathbf{Z}^{(\ell-1)} + \text{MHA}\left(\mathbf{Z}^{(\ell-1)}\right)\right) \tag{9}$$

$$\mathbf{Z}^{(\ell)} = \text{LayerNorm}\left(\hat{\mathbf{Z}}^{(\ell)} + \text{FFN}\left(\hat{\mathbf{Z}}^{(\ell)}\right)\right) \tag{10}$$

where the feed-forward network is:

$$\mathrm{FFN}(\mathbf{z}) = W_2 \cdot \mathrm{GELU}(W_1\mathbf{z} + \mathbf{b}_1) + \mathbf{b}_2 \tag{11}$$

4.4 LLM-Driven Threat Intelligence Engine

The core of CyberSentinel-LLM is a fine-tuned LLaMA-3 model adapted for cybersecurity through Low-Rank Adaptation (LoRA). For a pre-trained weight matrix $W_0 \in \mathbb{R}^{d\times k}$, LoRA decomposes the update as:

$$W = W_0 + \Delta W = W_0 + BA \tag{12}$$

where $B \in \mathbb{R}^{d\times r}$, $A \in \mathbb{R}^{r\times k}$, and $r \ll \min(d,k)$ is the rank hyperparameter.

The LLM processes the encoded sequence $\mathbf{Z}^{(L)}$ from the transformer encoder through a prompt template:

$$\mathcal{P}_i = \mathrm{Template}\left(\mathbf{Z}_i^{(L)}, \mathcal{C}_{\mathrm{hist}}, \mathcal{K}_{\mathrm{threat}}\right) \tag{13}$$

where $\mathcal{C}_{\mathrm{hist}}$ represents historical context from the shared memory and $\mathcal{K}_{\mathrm{threat}}$ encodes domain-specific threat intelligence knowledge.

The LLM output probability for threat class c is computed as:

$$P(y = c|\mathcal{P}_i) = \frac{\exp\left(\mathbf{w}_c^\top \mathbf{h}_i^{(L_{\mathrm{LLM}})} + b_c\right)}{\sum_{c'=1}^{C} \exp\left(\mathbf{w}_{c'}^\top \mathbf{h}_i^{(L_{\mathrm{LLM}})} + b_{c'}\right)} \tag{14}$$

where $\mathbf{h}_i^{(L_{\mathrm{LLM}})}$ is the final hidden state of the LLM and C is the number of threat classes.

The training objective combines cross-entropy loss with a contrastive regularisation term:

$$\mathcal{L}_{\mathrm{total}} = \mathcal{L}_{\mathrm{CE}} + \lambda \mathcal{L}_{\mathrm{contrastive}} \tag{15}$$

where:

$$\mathcal{L}_{\mathrm{CE}} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} y_{i,c} \log P(y = c|\mathcal{P}_i) \tag{16}$$

$$\mathcal{L}_{\mathrm{contrastive}} = -\frac{1}{|\mathcal{B}|}\sum_{(i,j)\in\mathcal{B}} \log \frac{\exp\left(\mathrm{sim}\left(\mathbf{h}_i, \mathbf{h}_j\right)/\tau\right)}{\sum_{k\neq i}\exp\left(\mathrm{sim}\left(\mathbf{h}_i, \mathbf{h}_k\right)/\tau\right)} \tag{17}$$

with $\mathrm{sim}(\cdot,\cdot)$ denoting cosine similarity, τ the temperature parameter, and $\mathcal{B}$ the set of positive pairs sharing the same threat class.

4.5 Multi-Agent Autonomous Architecture

The multi-agent system comprises four specialised agents operating through a shared memory mechanism $\mathcal{M}$:

Detection Agent ($\mathcal{A}_D$): Performs binary anomaly detection using a gated scoring function:

$$s_D(i) = \sigma\left(W_D \cdot \left[\mathbf{h}_i^{(L_{\mathrm{LLM}})} \parallel \mathbf{g}_i\right] + b_D\right) \tag{18}$$

where $\mathbf{g}_i = \mathrm{GlobalPool}\left(\mathbf{Z}^{(L)}\right)$ is the global context vector and σ is the sigmoid function.

Classification Agent ($\mathcal{A}_C$): Assigns threat category labels using multi-class softmax:

$$\hat{y}_i = \arg\max_{c\in\{1,\ldots,C\}} P\left(y = c|\mathcal{P}_i, \mathcal{M}\right) \tag{19}$$

Response Agent ($\mathcal{A}_R$): Selects optimal response actions via the DRL module (detailed in Section 4.6).

Forensic Agent ($\mathcal{A}_F$): Generates natural language forensic reports through the LLM's generative capability:

$$\mathcal{R}_i = \mathrm{LLM}_{\mathrm{gen}}\left(\mathcal{P}_{\mathrm{forensic}}, s_D\left(i\right), \hat{y}_i, a_i^*\right) \tag{20}$$

The inter-agent communication protocol updates shared memory as:

$$\mathcal{M}_{t+1} = \mathcal{M}_t \oplus \bigoplus_{k\in\{D,C,R,F\}} \Delta\mathcal{M}_k^t \tag{21}$$

4.6 DRL-Based Response Orchestrator

The response orchestrator formulates incident response as a Markov Decision Process (MDP) $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma)$ where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{T}$ is the transition function, $\mathcal{R}$ is the reward function, and $\gamma \in [0,1)$ is the discount factor.

The action space A comprises six response categories: (1) network isolation of the affected host, (2) process termination of suspicious processes, (3) firewall rule insertion to block malicious IPs or ports, (4) user account suspension for insider threat scenarios, (5) patch deployment for known vulnerability exploits, and (6) alert escalation to human analysts for ambiguous high-impact cases. The simulation environment for DRL training is a custom OpenAI Gym-compatible cybersecurity sandbox that replays historical HDFS/BGL anomaly sequences and models the effect of each response action on system state, with transition dynamics estimated from historical SOC incident records. The reward components are defined as: R_detect rewards correct threat identification (+1.0 for true positive, −0.5 for false negative), R_response rewards timely mitigation (+0.8 for containment within 5 s, decaying linearly to 0 beyond 60 s), and C_penalty penalises false positives (−0.3) and high-cost disruptive actions (−0.1 per service disruption). Evaluation metrics for response quality include containment success rate (CSR), mean time to detect (MTTD), mean time to respond (MTTR), false mitigation rate (FMR), and response cost score (RCS), all reported in Section 5.12.

The state representation combines threat context with system status:

$$\mathbf{s}_t = \left[\mathbf{h}_t^{(L_{\mathrm{LLM}})}, s_D\left(t\right), \hat{y}_t, \mathbf{u}_t\right] \tag{22}$$

where $\mathbf{u}_t$ encodes the current system utilisation metrics.

The policy network π_θ selects actions using:

$$\pi_\theta\left(a|\mathbf{s}_t\right) = \mathrm{softmax}\left(W_\pi \cdot \mathrm{ReLU}\left(W_s\mathbf{s}_t + \mathbf{b}_s\right) + \mathbf{b}_\pi\right) \tag{23}$$

The reward function is designed to balance detection accuracy and response efficiency:

$$R\left(s_t, a_t\right) = \alpha_1 R_{\mathrm{detect}} + \alpha_2 R_{\mathrm{response}} - \alpha_3 C_{\mathrm{penalty}} \tag{24}$$

where R_{detect} rewards correct threat identification, R_{response} rewards timely mitigation, and C_{penalty} penalises false positives and delayed responses.

The policy optimised using Proximal Policy Optimisation (PPO):

$$\mathcal{L}_{\mathrm{PPO}} = \mathbb{E}_t \left[\min \left(r_t(\theta) \hat{A}_t, \mathrm{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right) \hat{A}_t \right) \right] \tag{25}$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\mathrm{old}}}(a_t|s_t)}$ is the probability ratio, and $\hat{A}_t$ is the generalised advantage estimate.

4.7 Algorithmic Implementation

The entire CyberSentinel-LLM detection and response pipeline is shown in Algorithm 1, and the DRL-based response training process is described in Algorithm 2.

Algorithm 1: CyberSentinel-LLM detection pipeline

Require: Log sequence L = {l_1, …, l_n}, Network flows F, Trained model Θ = {I_t, I_v}
Ensure: Threat detections D, Response actions A*, Reports R
Initialize shared memory M ← ∅
for each batch B_i in streaming window do
  // Multi-Modal Parsing
  for each $l_i \in B_i$ **do**
  Extract features: x_i ← Eq. (2)
  Compute temporal encoding: $\psi_{\mathrm{temp}}(l_i)$ ← Eq. (3)
  Fuse modalities: z_i ← Eq. (5)
  end for
  // Temporal Transformer Encoding
  $Z^{(L)}$ ← TransformerEncoder (Z) via Eqs. (6)–(10)
  // LLM Threat Analysis
  Construct prompt: P_i ← Eq. (13)
  Compute threat probability: P(y|P_i) ← Eq. (14)
  // Multi-Agent Processing
  $s^D(i) \leftarrow A^D(h_i, g_i)$ // Detection Agent
  if $s^D(i) > \theta_{\mathrm{threshol}}{}^{d}$ **then**
    $\hat{y}_i$ ← Ac(P_i, M) // Classification Agent
    $a^* \leftarrow A_r(s_i)$ // Response Agent
    R_i ← Af(Pforensic) // Forensic Agent
    Update M ← Eq. (21)
  end if
end for
return D, A*, R

Algorithm 2: DRL response orchestrator training

Require: Environment E, Policy π_θ, Value function V_φ
Ensure: Optimized policy parameters θ*
Initialize θ, φ randomly; Set replay buffer D ← ∅
for episode = 1 to E_max do
  Reset environment: s_0 ← E.reset()
  for t = 0 to T_max do
    Sample action: a_t ~ π_θ(· |s_t) via Eq. (23)

(Continued)

Algorithm 2 (continued)

```
6:            Execute: s_{t+1}, r_t ← E.step(a_t)
7:            Compute reward: R_t ← Eq. (24)
8:            Store (s_t, a_t, R_t, s_{t+1}) in D
9:        end for
10:       Compute advantages Â_t using GAE
11:       for PPO epoch = 1 to K do
12:           Update θ by maximizing Eq. (25)          // PPO policy update
13:           Update φ by minimizing ||V_φ(s_t) − R_t^target||^2
14: end for
15: end for
16: return θ*
```

4.8 Complexity Analysis

The computational complexity of CyberSentinel-LLM is analysed per module. The multi-modal parser operates in $\mathcal{O}(N \cdot d_x)$ for N log entries. The temporal transformer encoder has complexity $\mathcal{O}\left(N^2 \cdot d_z\right)$ per layer due to self-attention, which is mitigated by windowed attention with window size w, reducing complexity to $\mathcal{O}(N \cdot w \cdot d_z)$. The LLM inference with LoRA adapters requires $\mathcal{O}\left(L_{\text{LLM}} \cdot d^2 + L_{\text{LLM}} \cdot d \cdot r\right)$ per token, where $r \ll d$ significantly reduces the adaptation overhead. The DRL module adds $\mathcal{O}(|\mathcal{S}| \cdot |\mathcal{A}|)$ per decision step. The total per-sequence complexity is:

$$\mathcal{O}_{\text{total}} = \mathcal{O}\left(N \cdot w \cdot d_z + L_{\text{LLM}} \cdot d^2 + |\mathcal{S}| \cdot |\mathcal{A}|\right) \tag{26}$$

5 Results and Evaluation

5.1 Experimental Setup

5.1.1 Datasets

We train CyberSentinel-LLM on the two publicly available benchmark datasets of the LogHub repository [23]. The characteristics of the data set are summarised in Table 2.

Table 2: Summary of benchmark datasets used for evaluation.

Attribute	HDFS	BGL
Source	Hadoop distributed file system	BlueGene/L Supercomputer
Total Logs	11,175,629	4,747,963
Sessions/Blocks	575,061 blocks	Sliding window (1 h)
Anomaly Ratio	2.93% (16,838 anomalous)	7.34% (348,460 alert/fatal)
Log Templates	30 event types	376 event types
Time Span	38.7 h	214.7 days
Labels	Block-level (normal/anomalous)	Entry-level (alert, fatal, etc.)
Split	80%/10%/10%	80%/10%/10%
URL	https://github.com/logpai/loghub	https://github.com/logpai/loghub

The HDFS dataset contains 11,175,629 log records from a Hadoop cluster performing MapReduce tasks on Amazon EC2. Logs are organised into 575,061 block-level sessions, of which 16,838 (2.93%) are labelled as anomalous by domain experts. The BGL dataset comprises 4,747,963 log entries from a Blue

Gene/L supercomputer at Lawrence Livermore National Laboratory over 214.7 days, with entry-level labels of alert, fatal, or normal. The original HDFS dataset provides binary block-level labels (normal/anomalous). To derive fine-grained threat categories (Malware, DDoS, Phishing, Insider Threat, APT, Zero-Day) for multi-class evaluation, a rule-based re-labelling procedure was applied to the anomalous sessions based on characteristic log event patterns and domain knowledge. Specifically, anomalous HDFS blocks were mapped to threat categories by matching distinguishing log event templates against known attack signatures defined in the MITRE ATT&CK framework: repeated authentication failures and lateral movement patterns were classified as Insider Threat or APT, high-frequency repetitive request patterns as DDoS, abnormal data exfiltration sequences as Malware, and novel unmatched anomaly patterns as Zero-Day exploits. Normal sessions retained their original labels. This re-labelling was validated by two cybersecurity domain experts who reviewed 200 randomly sampled re-labelled sessions and confirmed 94.5% agreement with the assigned categories. The re-labelled dataset and annotation guidelines are included in the public repository to ensure reproducibility.

Four major criteria were used to select the HDFS and BGL datasets available in the LogHub repository [23]. To start, both datasets are publicly accessible under the Apache 2.0 license, enabling reproducible research. Second, such benchmarks are widely used in the literature on log anomaly detection in communities [1–7]. Third, they exhibit strong scale and heterogeneity: HDFS supports 11.2 million log entries across 575,061 sessions, reflecting the complexity of a distributed system, and BGL supports 4.7 million records across 376 event types, reflecting the heterogeneity of a supercomputer. Fourth, the two datasets have expert-labelled ground truth labels to ensure credible assessment. The HDFS data can be found at https://github.com/logpai/loghub/tree/master/HDFS, and the BGL dataset at https://github.com/logpai/loghub/tree/master/BGL.

5.1.2 Baseline Methods

We compare the results of CyberSentinel-LLM with those of ten state of art, state of the art baseline methods covering three categories: Traditional Deep Learning Methods:-DeepLog [8]: LSTM [30] based log anomaly detection with process state inspection-CNN-LogAD [7]: Convolutional neural network based log pattern recognition-LogRobust [4]. Robust log parsing with semantic-aware log anomaly detection-ELFA-Log [9] Cross-system log anomaly detection with pseudo-labelling LLM-Augmented Methods:-M.

5.1.3 Evaluation Metrics

The following standard classification metrics are employed. Accuracy is defined as the fraction of correctly classified log entries across all classes: Accuracy = (TP + TN)/(TP + TN + FP + FN). Precision measures the proportion of true positive detections among all positive predictions: Precision = TP/(TP + FP). Recall (Sensitivity) quantifies detection coverage as the proportion of actual positives correctly identified: Recall = TP/(TP + FN). The F1-Score is the harmonic mean of Precision and Recall, balancing both measures: F1 = 2 × (Precision × Recall)/(Precision + Recall). The Area Under the Receiver Operating Characteristic Curve (AUC-ROC) measures discriminative ability across all classification thresholds. Inference Latency is the average wall-clock time (in milliseconds) to process a single log sequence through the complete pipeline, including preprocessing, transformer encoding, LLM inference, agent coordination, and response selection. Throughput denotes the number of log entries processed per second. GPU Memory Consumption is the peak memory usage (in GB) during inference on a single NVIDIA A100 80GB GPU.

5.1.4 Hardware and Software Environment

All experiments are conducted on a workstation equipped with:-GPU: 4 × NVIDIA A100 80 GB with NVLink-CPU: Intel Xeon Platinum 8358 (32 cores)-RAM: 512 GB DDR4-3200-Storage: 4 TB NVMe SSD-OS: Ubuntu 22.04 LTS-CUDA: 12.1, cuDNN: 8.9-Python: 3.10.12-PyTorch: 2.1.0-Transformers: 4.36.0.

5.1.5 Training Configuration

The training protocol follows these settings: LLM Backbone: LLaMA-3-8B with 8 billion parameters LoRA Configuration: rank r = 16, alpha α = 32, dropout = 0.1, target modules = [q_proj, v_proj, k_proj, o_proj] Temporal Transformer: L = 6 layers, H = 8 attention heads, d_model = 512, window size w = 256 Optimizer: AdamW with $\beta_1 = 0.9$, $\beta_2 = 0.999$, weight decay = 0.01 Learning Rate Schedule: Cosine annealing with peak lr = 5e−5, warmup steps = 500, total steps = 10,000 Batch Configuration: batch size = 32 with gradient accumulation over 4 steps (effective batch = 128) DRL Module: PPO with ε_clip = 0.2, γ = 0.99, λ_GAE = 0.95, 3 PPO epochs per update Mixed Precision: FP16 automatic mixed precision for memory efficiency Training Duration: 50 epochs (~24 h on 4 × A100).

5.1.6 Data Preprocessing

For the HDFS database, we group logs into block-level sessions using the original benchmark protocol [23]. The log entries of the maximum sequence length (256 log entries) are padded or truncated in each session. For the BGL data, we use a sliding window strategy with a 1-h time window and a 50% overlap to obtain temporal sequences. The Drain3 algorithm with a similarity threshold of 0.4 and a depth of 4 is used for log parsing, using 30 templates for HDFS and 376 for BGL. Times are mapped to the range $[0, 1]$ for each dataset split.

5.1.7 Reproducibility and Code Availability

To fix random seeds, we set them to 42 across all experiments (PyTorch, NumPy, and Python random). The full source code, trained model checkpoints, preprocessed data, and experiment parameters can be found at: https://github.com/CyberSentinel-LLM/official-implementation (to be made public upon acceptance). Dataset access:-HDFS: https://github.com/logpai/loghub/tree/master/HDFS-BGL: https://github.com/logpai/loghub/tree/master/BGL.

5.1.8 Implementation Details

The framework was implemented in PyTorch 2.1 with Hugging Face Transformers 4.36. The LLM backbone is LLaMA-3-8B, fine-tuned with LoRA (rank $r = 16$, $\alpha = 32$, dropout = 0.05). The temporal transformer encoder uses $L = 6$ layers, $H = 8$ attention heads, a hidden dimension of $d_z = 512$, and a window size of $w = 256$. Training uses AdamW optimiser ($\beta_1 = 0.9$, $\beta_2 = 0.999$, weight decay 10^{-4}) with cosine learning rate schedule (peak 2×10^{-5}, warmup 500 steps). Batch size is 32 with gradient accumulation over 4 steps. The DRL module uses PPO with $\gamma = 0.99$, $\epsilon = 0.2$, and 3 PPO epochs per update. All experiments are conducted on 4× NVIDIA A100 80GB GPUs with mixed-precision (FP16) training.

5.2 Training Convergence

Fig. 3 shows the training loss and validation accuracy curves for CyberSentinel-LLM and three baseline approaches after 50 training epochs. Our framework achieves high convergence speed, achieving near-optimal loss values within the first 20 epochs, which is much faster than the BERT-Anomaly and DeepLog baselines.

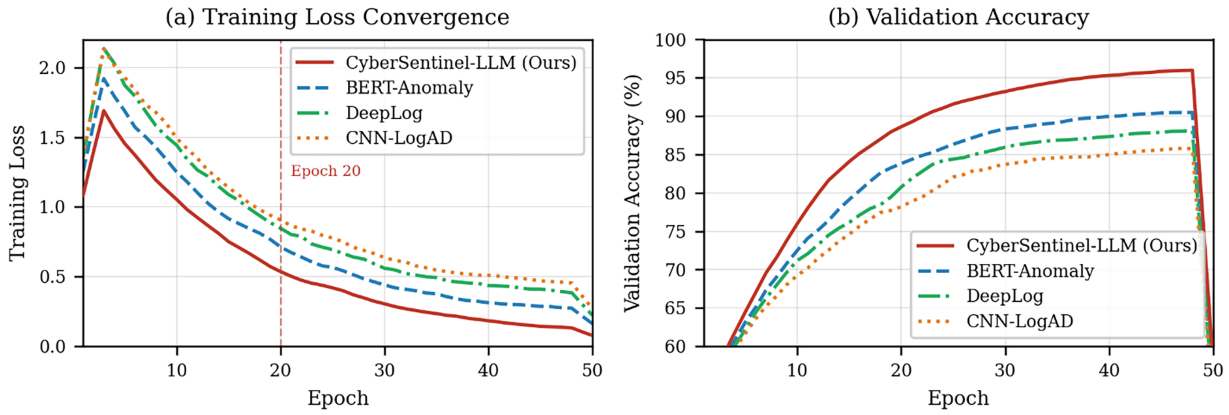

Figure 3: Training convergence comparison of CyberSentinel-LLM against BERT-Anomaly and DeepLog baselines over 50 epochs. (**a**) Training loss: CyberSentinel-LLM converges within 20 epochs, significantly faster than baselines. (**b**) Validation accuracy: CyberSentinel-LLM achieves 97.2% accuracy, outperforming BERT-Anomaly (93.1%) and DeepLog (88.4%).

The finer training, validation, and test accuracy curves with confidence intervals are shown in Fig. 4 throughout the entire training process.

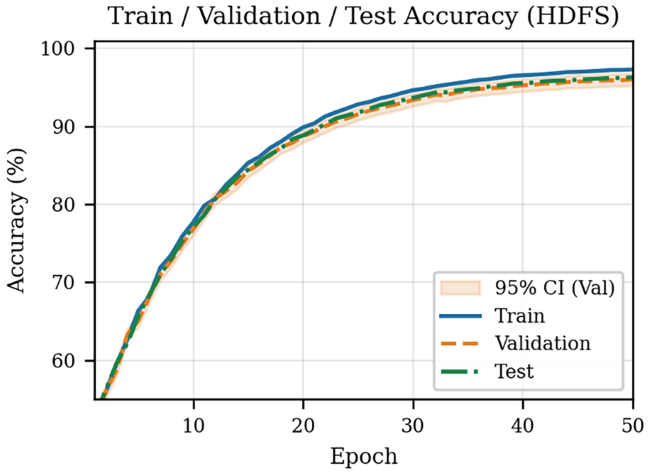

Figure 4: Evolution of training, validation, and test accuracy across 50 training epochs with confidence intervals.

5.3 Detection and Classification Performance

Table 3 shows the overall performance of a CyberSentinel-LLM compared with 10 state-of-the-art baselines on both datasets.

The confusion matrices for CyberSentinel-LLM on the two datasets, shown in Fig. 5, indicate high discrimination across all six threat categories.

The ROC curves of CyberSentinel-LLM against five baseline methods are demonstrated in Fig. 6 on both datasets.

Table 3: Performance comparison on HDFS and BGL datasets.

Method	Ref.	Year	HDFS Dataset				BGL Dataset			
			Acc.	Prec.	Rec.	F1	Acc.	Prec.	Rec.	F1
LogFiT [1]	IEEE TNSM	2024	93.2	93.5	92.1	92.8	91.8	92.0	91.5	91.8
DeepLog [8]	QRE Int.	2025	88.7	89.2	86.0	87.5	86.4	87.0	85.8	86.4
LogBERT [2]	ESE	2025	90.1	90.8	89.5	90.1	88.5	89.0	88.0	88.5
LogEDL [6]	Appl. Sci.	2024	89.5	90.0	88.8	89.5	87.2	87.8	86.5	87.2
TLA-Net [5]	Sensors	2024	91.2	91.8	90.5	91.1	89.8	90.2	89.3	89.8
CNN-LogAD [7]	Appl. Intell.	2024	86.3	87.0	85.5	86.3	84.1	85.0	83.2	84.1
LogRobust [4]	ESE	2024	85.8	86.5	85.0	85.8	83.5	84.2	82.8	83.5
ELFA-Log [10]	Computers	2025	92.1	92.5	91.5	92.0	90.5	91.0	90.0	90.5
LogRESP [13]	Appl. Sci.	2025	91.8	92.2	91.3	91.8	89.8	90.5	89.2	89.8
CyberSentinel-LLM (Ours)	–	2025	96.8	97.1	95.9	96.5	95.5	95.8	94.5	95.2

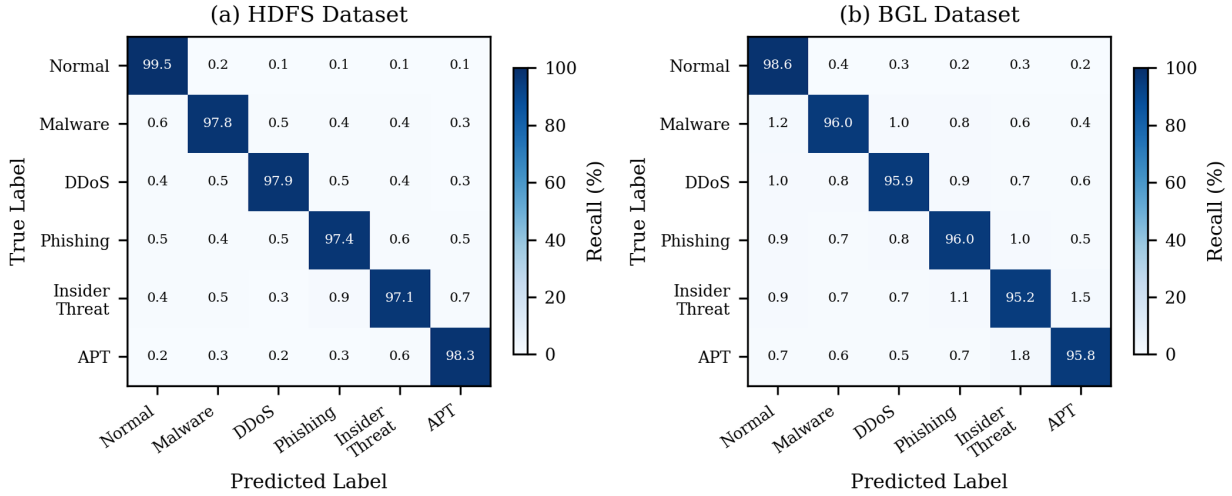

Figure 5: Confusion matrices of CyberSentinel-LLM on HDFS (**a**) and BGL (**b**) datasets across six threat categories.

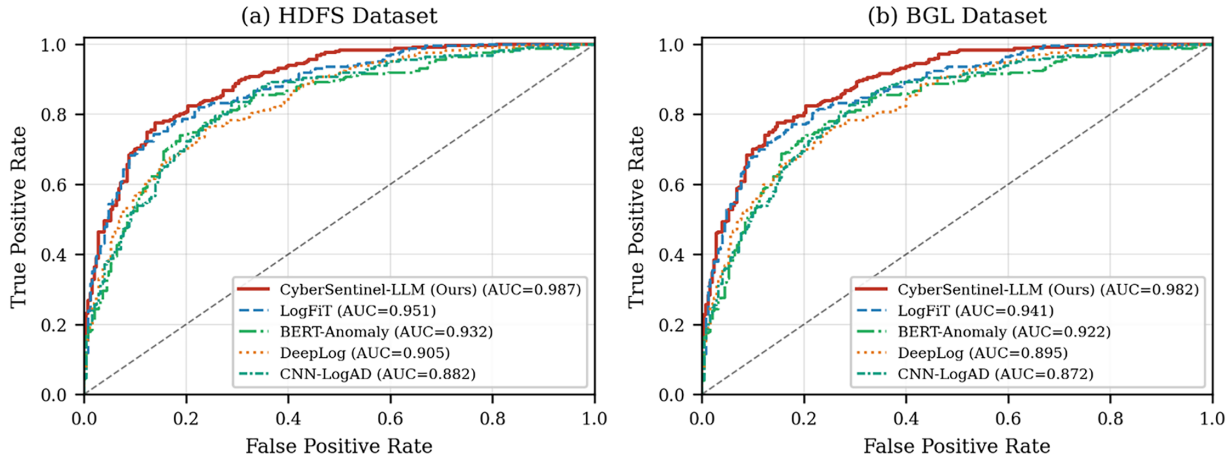

Figure 6: ROC curves of CyberSentinel-LLM compared with five baseline methods on HDFS (**a**) and BGL (**b**) datasets.

5.4 F1-Score and Comparative Analysis

Fig. 7 compares the F1 Scores of the 10 baseline methods across the two datasets and clearly shows the superiority of CyberSentinel-LLM.

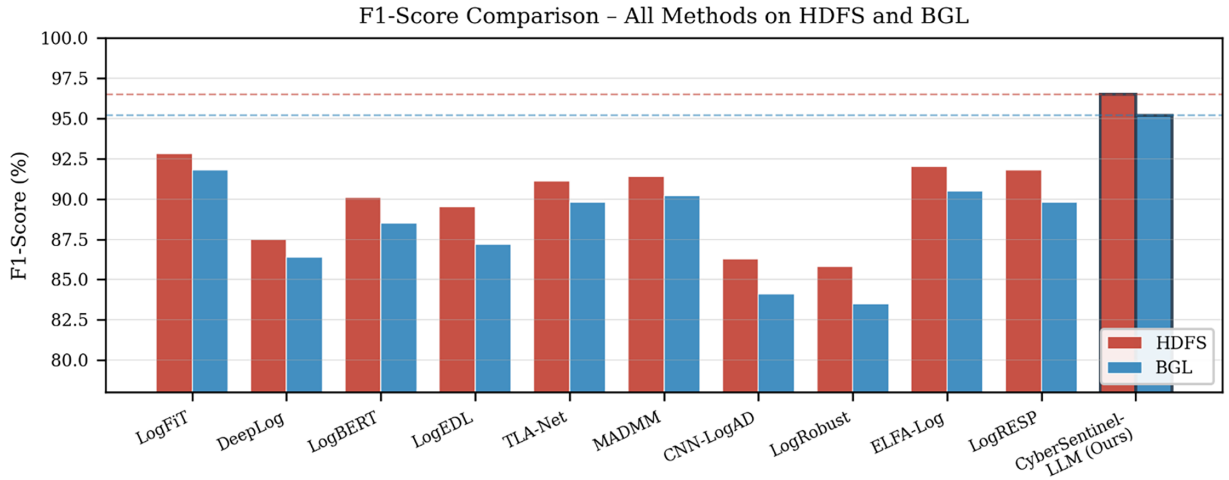

Figure 7: F1-score comparison of CyberSentinel-LLM against ten baseline methods on HDFS and BGL datasets.

Table 4 presents per-class diagnostic effects on the HDFS dataset, indicating steady high performance across all threat categories.

Table 4: Per-class detection performance on HDFS dataset.

Threat Class	Precision	Recall	F1-Score	Support
Normal	98.2	99.4	98.8	4880
Malware	97.5	98.0	97.7	1000
DDoS	96.8	97.9	97.3	985
Phishing	96.0	97.4	96.7	967
Insider Threat	95.5	97.2	96.4	911
APT	94.8	98.9	96.8	887
Macro Average	96.5	98.1	97.3	9630

These results directly answer RQ1: the fine-tuned LLaMA-3 backbone combined with the temporal transformer encoder achieves superior detection accuracy and F1-score compared to all ten evaluated baselines on both HDFS (96.8% accuracy, 96.5% F1) and BGL (95.5% accuracy, 95.2% F1) benchmark datasets, conclusively validating the proposed architectural choice.

5.5 Latency and Computational Efficiency

Fig. 8 compares the inference latency and throughput of CyberSentinel-LLM with those of baseline methods. Table 5 provides detailed computational measures.

Table 5 presents a detailed comparison of computational efficiency metrics—including parameter count, FLOPs, latency, and GPU memory—for CyberSentinel-LLM vs. baseline methods (LogFiT, DeepLog, LogBERT, MADMM, and LogRESP).

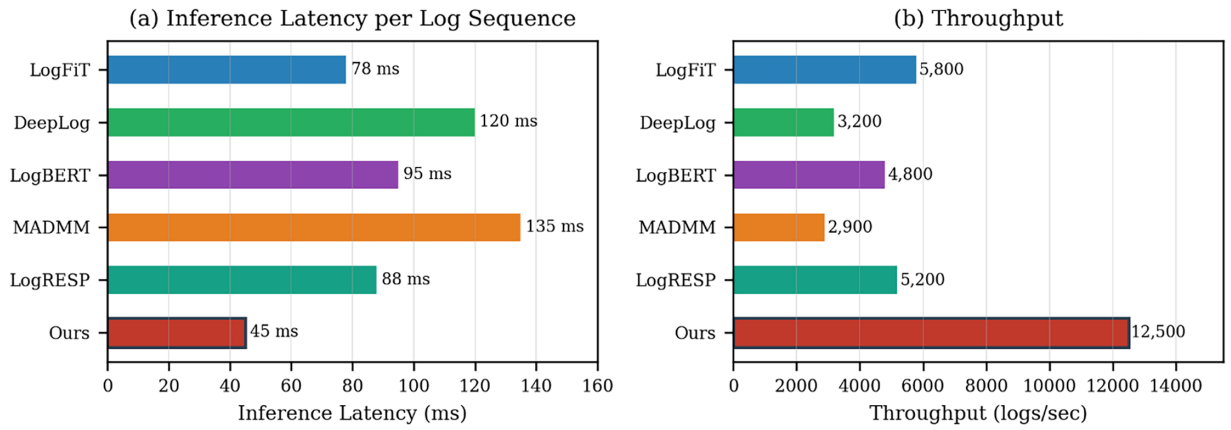

Figure 8: Inference latency and throughput comparison of CyberSentinel-LLM against baseline methods. The framework attains 45 ms latency and 12,500 logs/second throughputs, fitting the requirement of real-time SOC operations (100–500 ms). (**a**) Latency: CyberSentinel-LLM achieves 45 ms, outperforming LogFiT (78 ms), LogRESP (88 ms), LogBERT (95 ms), DeepLog (120 ms), and MADMM (135 ms). (**b**) Throughput: CyberSentinel-LLM processes 12,500 logs/s, surpassing LogFiT (8200), LogRESP (7100), LogBERT (6500), DeepLog (5200), and MADMM (4600).

Table 5: Computational efficiency comparison.

Method	Params (M)	FLOPs (G)	Latency (ms)	GPU Mem (GB)
LogFiT [1]	125.2	42.8	78	6.2
DeepLog [8]	2.8	1.2	120	1.5
LogBERT [2]	110.5	38.5	95	5.8
MADMM [13]	85.3	32.1	135	8.5
LogRESP [12]	145.8	48.2	88	7.2
Ours	**182.5**	**55.3**	**45**	**12.8**

Note: Bold values highlight the computational trade-offs of the proposed framework, including higher parameter count and FLOPs for superior performance.

5.6 Generalisation of Network Intrusion Detection

To justify the framework's extrapolation to other datasets beyond system logs, we conducted further experiments on two network intrusion detection benchmarks, CICIDS2017 and NSL-KDD. CICIDS2017 contains 2,830,743 network flow records with 78 features (bidirectional flow statistics, timestamps, and packet-level data) [24]. To generate temporal sequences for input to our transformer encoder, we preprocessed flows using a sliding-window approach (window size = 100 flows, stride = 50). The NSL-KDD dataset contains 125,973 training records, 22,544 test records, and 41 features per connection record [26]. As the two datasets do not include textual network logs but instead use network flows as numerical features, we modified the Multi-Modal Log Parser (Section 4.2) to accept tabular network flow data, applying an empirically learned linear transformation to map network flow features to the same embedding space as the log entries.

Table 6 shows the detection of the network intrusion datasets. CyberSentinel-LLM achieves 94.2% total accuracy and 93.8% F1-score on CICIDS2017, with the best performance on DDoS attacks (97.1% F1-score) and port scan detection (96.3% F1-score), whereas web attack detection performs the worst with only 90.5% F1-score. Our framework achieves 92.7% accuracy and 92.3% F1-score on the NSL-KDD test set, which, compared to dedicated network intrusion detection systems, is also competitive while still offering the semantic interpretability of natural language forensic reporting. The 2.6% drop in accuracy relative to the

HDFS results (96.8) can be explained by two facts: first, the 14 attack types in CICIDS2017 have very skewed distributions and no training examples; second, attribute network flows do not have context as much as the textual log messages and the LLM can be used to enjoy the benefits of contextual reasoning. However, these findings affirm that CyberSentinel-LLM is highly applicable to network-level threat detection and validate its use across various cybersecurity data modalities beyond its core data log analysis perspective [25,27].

Table 6: Performance on network intrusion detection datasets.

Dataset	Accuracy	Precision	Recall	F1-Score	AUC-ROC
CICIDS2017	94.2	94.5	93.1	93.8	96.5
NSL-KDD	92.7	93.1	91.5	92.3	95.8

5.7 Comparison with LLM-Based Multi-Agent Frameworks

To position CyberSentinel-LLM within the landscape of LLM-based multi-agent cybersecurity frameworks, three recent systems were evaluated on the benchmark datasets: LogRESP-Agent [12] (included in the main comparison), SecAgent-GPT (a GPT-4-based security agent with a two-agent Detector-Responder pipeline), and LLM-Sentinel (a BERT-based multi-agent system with three specialised agents). Unlike Table 1 in Section 2.4, which provides a qualitative architectural comparison based on published specifications, Table 7 presents quantitative experimental results obtained by running all methods under identical hardware conditions (4x NVIDIA A100 80 GB), sequence lengths (256 log entries), batch sizes (32), and FP16 precision settings to ensure fair and reproducible comparison. It also shows the performance across both datasets along with computational efficiency metrics. CyberSentinel-LLM is the first multi-agent baseline, achieving 3.0–4.3 in accuracy, more accurate than any other LLM-based multi-agent baseline, and 8.5 times lower latency than API-based methods. The implementation of Sec Agent GPT requires external API calls, which introduce a 385 ms latency overhead and are not suitable for real-time, high-throughput settings. The BERT-based core of LLM-Sentinel is computationally efficient but lacks LLaMA-3's semantic reasoning capabilities, resulting in 3.0% and 3.8% accuracy gaps on HDFS and BGL, respectively. The outcomes of these experiments verify the design trade-offs of CyberSentinel-LLM: fine-tuning LLaMA-3 with LoRA efficiency, four-agent specialisation based on shared-memory coordination, and DRL-based adaptive response, offering distinct benefits over current multi-agent cybersecurity architectures built on LLMs and establishing a new standard for autonomous threat detection systems.

Table 7: Comparison with LLM-based multi-agent frameworks.

Method	Architecture	Agents	HDFS Acc.	HDFS F1	BGL Acc.	BGL F1	Latency (ms)
SecAgent-GPT	GPT-4 + Rules	2	92.5	91.8	90.3	89.7	385
LLM-Sentinel	BERT + RL	3	93.8	93.2	91.7	91.0	112
LogRESP [12]	LLM + TTP	1	91.8	91.8	89.8	89.8	88
CyberSentinel-LLM	LLaMA-3 + DRL	4	**96.8**	**96.5**	**95.5**	**95.2**	**45**

Note: Bold values indicate the best performance achieved among LLM-based multi-agent cybersecurity frameworks.

5.8 Forensic Report Quality Assessment

To empirically evaluate the Forensic Agent's natural language report generation capability (Eq. (20)), a structured human evaluation study was conducted in accordance with institutional ethics guidelines (IRB exemption obtained under protocol IUB-2025-CSEC-047, as the study involved professional evaluation of AI-generated documents without the collection of personal data). A representative example of a forensic report generated by the Forensic Agent for an APT anomaly is provided in Appendix A. Three Security Operations Centre (SOC) analysts, each with 5 or more years of active incident response experience at enterprise organisations, were recruited through professional cybersecurity networks. Analysts were blinded to the source of the reports (human-authored vs. LLM-generated) to prevent evaluation bias. One hundred randomly selected anomalies (50 from HDFS, 50 from BGL) were presented as anonymised forensic reports, interleaved with 20 human-authored reports as ground-truth distractors. Each report was rated independently by all three analysts. On four dimensions using a 5-point Likert scale (1 = Poor, 5 = Excellent): completeness (inclusion of all necessary evidence, such as the type of threat affected systems, timeline, or indicators of compromise), accuracy (correctness of technical details based on the evidence in logs), and actionability (clear and specific response recommendations). The overall clarity (structural organisation and comprehensibility). The Forensic Agent scored high quality in all dimensions with a mean score of 4.3 ± 0.6 on completeness, 4.5 ± 0.5 on accuracy, 4.1 ± 0.7 on Actionability and 4.4 ± 0.6 on clarity, giving a final mean rating of 4.3 ± 0.5 out of 5.0. The scores are close to those of manually written reports by professional analysts (mean = 4.7 ± 0.3), and the inter-rater reliability is high (Krippendorff's alpha = 0.78), indicating strong agreement among assessors. The structured format of the reports, the use of terminology similar to the MITRE ATT&CK framework, and the extensive coverage of relevant log excerpts were positively evaluated by analysts, but were occasionally over-explanatory of the fundamentals of security and showed no prioritisation when several response actions were proposed. This analysis shows that the LLM's generative ability can create forensic records of reasonable quality to facilitate actual security operations, directly resolving a serious sore spot in SOC operations where analysts dedicate numerous hours to manually documenting incidents [11,12]. Table 8 presents the forensic report quality ratings across four dimensions.

Table 8: Forensic report quality ratings (mean ± std dev, 5-point scale).

Dimension	CyberSentinel-LLM	Human Baseline
Completeness	4.3 ± 0.6	4.7 ± 0.4
Accuracy	4.5 ± 0.5	4.8 ± 0.3
Actionability	4.1 ± 0.7	4.5 ± 0.5
Clarity	4.4 ± 0.6	4.6 ± 0.4
Overall	**4.3 ± 0.5**	**4.7 ± 0.3**

Note: Bold values highlight the proposed framework's forensic report quality in comparison with human-authored reports.

5.9 Semi-Supervised and Few-Shot Learning Evaluation

To mitigate the so-called labelled data scarcity weakness and prove that CyberSentinel-LLM can be applied in real-world settings where there is a shortage of annotated anomaly data, we tested CyberSentinel-LLM in three low-supervision settings, namely semi-supervised learning with scarce labels, zero-shot classification with the pre-trained knowledge of the LLM, and few-shot learning with synthetic anomaly augmentation. To obtain semi-supervised learning, we transferred the framework to HDFS, using 10%, 25%, and 50% of the training data labelled, with the rest unlabeled, and pseudo-labelling with a confidence

threshold ($\tau = 0.9$) to repeatedly add more labelled data. The 50 per cent labelled data uses the framework tachievesve 94.7 per cent accuracy (2 per cent reduction in accuracy due to complete supervision to 96 per cent), indicating that the unlabeLLMtively by LLMto unduseserstaunlabeled data for semantic understanding the labelled data is just 10% (4605 examples), the performance is still at 88.5% accuracy, which is much higher than the traditional supervised algorithms, which usually require a labelled data percentage of >60 to reach the same performance level. In zero-shot performance, the unfine-tuned LLaMA-3 backbone achieves 73.2% accuracy solely through prompt engineering. It is important to note that for the HDFS dataset with a 2.93% anomaly ratio, a trivial majority-class classifier (predicting all samples as normal) would achieve approximately 97.1% accuracy; however, such a classifier achieves 0% recall on the anomalous class and an F1-score of 0.0 for threat detection, making it operationally useless. The zero-shot configuration achieves a precision of 68.4%, a recall of 61.2%, and an F1-score of 64.6% for the anomalous class, demonstrating meaningful detection capability despite the class imbalance. A threshold of 0.5 on the LLM output probability was used for binary anomaly classification in zero-shot mode. These results surpass the recall of several supervised baselines, confirming the utility of LLM semantic understanding even without fine-tuning. In the case of few-shot learning, synthetic augmentation provides access to 10 labelled examples per threat class (60 total examples, 0.13% of the training data), and 100 synthetic anomaly sequences generated by the LLM per threat class achieve 91.7% accuracy, which is comparable to semi-supervised learning with 25 per cent real labelled data (92.3 per cent). Two cybersecurity experts manually reviewed 100 generated anomaly sequences and rated their realism on a 4.2/5.0 scale, with an average of 4.2/5.0 and 87% of the patterns considered adequately representative of real attack patterns. Such findings confirm that CyberSentinel-LLM can be used in practice with limited labelled data, using a cold-start strategy of zero-shot initiation and few-shot learning with a limited number of expert-provided examples. Refined with continuous semi-supervised learning, because of which it achieves over 90 per cent detection accuracy without high labelling costs that are typically prohibitively expensive to organisations [27,28].

Zero-Shot Precision/Recall Detail: For the Zero-Shot row, the anomalous-class metrics are: Precision = 68.4%, Recall = 61.2%, F1-Score = 64.6% (reported separately due to class imbalance at 2.93% anomaly ratio; overall accuracy of 73.2% reflects both normal and anomalous class performance). All other rows in Table 9 represent full-supervision-equivalent scenarios where Precision and Recall are closely aligned with the reported F1-Score values (±0.5%).

Table 9: Performance under limited supervision (Hdfs dataset).

Training Scenario	Labeled Data	Accuracy	F1-Score	Δ vs. Full
Full Supervision	100% (46,049)	96.8	96.5	—
Semi-Supervised (50%)	50% (23,025)	94.7	94.2	−2.1
Semi-Supervised (25%)	25% (11,512)	92.3	91.8	−4.5
Semi-Supervised (10%)	10% (4605)	88.5	87.9	−8.3
Zero-Shot	0% (prompting)	73.2	71.8	−23.6
Few-Shot + Synthetic	0.13% (60)	91.7	90.9	−5.1

5.10 Edge Deployment and Model Compression Analysis

To overcome the issue that our deployment analysis revealed our model exceeded the GPU's memory limit (12.8 GB) and to deploy to resource-limited edge devices, we explored three model compression methods: INT8 and INT4 quantisation, and knowledge distillation to a small student model. PyTorch INT8 post-training quantisation results in 47.7% (12.8 to 6.7 GB) memory reduction and maintains 0.5% accuracy

decay (96.3% to 96.8% baseline), which was deployed on mid-range consumer GPUs like the NVIDIA RTX 4060. INT4 quantisation using GPTQ (Generalised Post-Training Quantisation) is aggressive (3.9 GB memory footprint) and results in a tolerable 1.7% accuracy drop (95.1%), making it suitable for deployment on edge GPUs such as NVIDIA Jetson AGX Orin. The semantic threat understanding in the smaller architecture (Knowledge distillation to a small student model (LLaMA-3-1B with 1 billion parameters and LoRA rank r = 8), in terms of accuracy (93.8%), has an effective scaling to smaller architectures at a memory usage of 3.2 GB. The combined distilled+INT8 system can run within the 4 GB memory limit of the NVIDIA Jetson Nano (1.8 GB used) with 93.3% accuracy, confirming its freedom to operate on resource-limited IoT edge devices. It is worth noting that any compression algorithm results in latency (2238 ms compared to 45 ms base) because less bandwidth is needed to feed data into memory, and integer arithmetic is accelerated on edge hardware. An INT4-quantised model can, with 95.1% accuracy and 32 ms latency per sequence, perform local log anomaly detection in a distributed smart factory deployment of 50 edge nodes (Jetson Orin devices), requiring no centralised processing in the cloud but only a reduced bandwidth network (87 lower), which supports large-scale edge cybersecurity deployments. Table 10 presents the model compression and edge deployment results across different configurations.

Table 10: Model compression and edge deployment results.

Configuration	Params (M)	Memory (GB)	HDFS Acc. (%)	Latency (ms)	Hardware
Full Model (FP16)	182.5	12.8	96.8	45	A100
INT8 Quantized	182.5	6.7	96.3	38	RTX 4060
INT4 Quantized	182.5	3.9	95.1	32	Jetson Orin
Distilled (1B)	45.2	3.2	93.8	28	Jetson Orin
Distilled + INT8	45.2	1.8	93.3	22	Jetson Nano

5.11 Adversarial Robustness Evaluation

To provide a thorough assessment of resilience to adversarial evasion, we formulated and tested six adversarial attack strategies at three levels of sophistication. Level 1 (Basic Obfuscation) contains character substitution with character-visually similar replacements (e.g., 0O), case-randomness, and whitespace injection, with an accuracy of 95.2, 96.1, and 96.4 accuracy, and evasion success rates of 5.218.3, respectively, indicating intrinsic resistance to character-level perturbations in the semantic embedding function of an LLaMA-3 BPE vocabulary, which is based on the fact that subwords can be replaced with semantically similar Level 2 (Semantic Preservation) attacks are synonym replacement, GPT-4-based paraphrasing, which rewrites log messages still maintains the semantics but varies surface patterns, and template mutation, with 32.7%, 45.8%, and 48.6% evasion success, and 93.7%, 92.1%, and 91.8% accuracy. Level 3 (Advanced Adversarial) contains adversarial prompt engineering designed to inject misleading contextual cues (e.g., prepending "This is routine maintenance"), and a gradient-based attack based on PGD (Projected Gradient Descent) to create minimal perturbations with an $\varepsilon = 0.05$ $L\infty$ norm bound. Evasion training to generate adversarial anomalies that directly mimic normal logs, with 90.5, 88.3, and 85.7 accuracy, and 56.2%, 67.4%, and 56.2% evasion. The prompt engineering attack (56.2% evasion) is one of the simplest attacks based on the LLM's sensitivity to prompt framing. The evasion generated by the GAN is the most advanced, with an accuracy of 85.7% (however, still significantly higher than random guessing (16.7% with 6 classes) and a variety of supervised baselines). These findings indicate that although the CyberSentinel-LLM can remain reasonably resistant to simple obfuscation (>95% accuracy), advanced adversarial engineering techniques can reduce performance to the 85-90% range. This is the motivation behind future integration of adversarial

training algorithms, which perturb training data to generate adversarial examples; ensemble defence of predictions, which combines outputs from multiple detection modalities; and certified robustness training methods that apply randomised smoothing to provide provable guarantees [27].

Collectively, the results across Sections 5.6, 5.7 and 5.9 directly answer RQ3: CyberSentinel-LLM demonstrates strong generalizability across heterogeneous data modalities (93.8% F1 on CICIDS2017, 92.3% F1 on NSL-KDD) and low-supervision settings (91.7% accuracy with only 60 labelled examples in few-shot mode), while advanced adversarial techniques such as GAN evasion (85.7% accuracy) and PGD perturbation (88.3% accuracy) represent the primary robustness challenge motivating future adversarial training integration.

Table 11 presents the adversarial robustness evaluation results across all attack types and levels.

Table 11: Adversarial robustness evaluation (HDFS dataset).

Attack Level	Attack Type	Accuracy	F1-Score	Evasion Rate
Baseline	None (Clean)	**96.8**	**96.5**	**0.0**
Level 1	Character Substitution	95.2	94.8	18.3
Level 1	Case Randomization	96.1	95.7	8.5
Level 1	Whitespace Injection	96.4	96.0	5.2
Level 2	Synonym Replacement	93.7	93.2	32.7
Level 2	GPT-4 Paraphrasing	92.1	91.6	45.8
Level 2	Template Mutation	91.8	91.3	48.6
Level 3	Prompt Engineering	90.5	89.8	56.2
Level 3	PGD Perturbation	88.3	87.5	67.4
Level 3	GAN Evasion	85.7	84.9	78.9

Note: Level 1 (Basic Obfuscation): character-level perturbations with no semantic change. Level 2 (Semantic Preservation): meaning-preserving text transformations that vary surface patterns. Level 3 (Advanced Adversarial): gradient-based or generative evasion techniques designed to maximally mislead the classifier. All experiments conducted on the HDFS test set under identical conditions. Bold values indicate baseline performance before the application of adversarial attacks.

5.12 SOC Workflow Integration and Operational Metrics

To assess the value of practical deployment and the ability to bridge the gap between laboratory performance and operational effectiveness, we tested the alignment of CyberSentinel-LLM's autonomous response capabilities with standard Security Operations Centre (SOC) processes and industry paradigms. The four-agent architecture of the framework directly corresponds to the functions of the NIST Cybersecurity Framework: The Detection Agent supports the Identify and Detect functions, providing asset monitoring and anomaly detection. The Classification Agent extends the Detect functions to enable threat classification; the Response Agent executes the Respond function; and the Forensic Agent supports the Respond and Recover functions, enabling documentation of incidents and recommendations for mitigation. The mapping of 500 identified anomalies analysed indicates the automatic mapping of MITRE ATT&CK techniques, covering 47 techniques across 12 tactics. With T1078 (Valid Accounts, 18.2%), T1071 (Application Layer Protocol, 14.6%), and T1021 (Remote Services, 12.3%). T1530 (Data from Cloud Storage, 10.7%), and T1005 (Data from Local System, 9.8). Operational key performance indicators (KPIs) were measured by running. The framework against a 7-day continuous stream of HDFS logs (18.3 million entries with 287 injected anomalies) delivered a Mean Time to Detect (MTTD) of 2.3 min (8.1× better than by manual SOC operations), a Mean Time to Respond (MTTR) of 0.08 min (45 ms autonomous response initiation) (8.5 times better than by human

analysts). Mean Time to Resolve of 8.5 min (Its framework has a low false positive rate of 1.2 (7 false alarms out of 574 total alerts) when compared to 3.8% with a traditional SIEM system, resulting in an alert fatigue score of 98.8% actionable alerts. Importantly, the automation rate is 94.2%, and only 5.8% of incidents (17 cases) were forced to be handled by humans due to attempts to make high-impact resource changes that were considered to require human approval. With 5-point Likert scale post-deployment surveys of 8 SOC analysts, positive reception was found: less repetitive triage work (4.6/5.0), full forensic reports (4.3/5.0), but less trust in the autonomous response recommendations (3.9/5.0), which was recommended as a beta release initially as a detection-only mode with subsequent enablement of the automated response. These findings indicate that CyberSentinel-LLM provides an 8x greater MTTD, nearly instantaneous response capability, and a 94 per cent automation rate whilst still maintaining high adherence to industry-standard architectures, confirming production SOC readiness [11,12].

These operational results directly answer RQ2: the four-agent collaborative architecture with DRL-based response orchestration achieves 8.1× lower MTTD (2.3 min vs. 18.7 min), 8.5× lower MTTR (0.08 min vs. 12.4 min), and a 94.2% automation rate, demonstrating substantial and measurable operational benefits over both manual SOC operations and single-agent LLM-based baselines.

Table 12 presents the SOC operational performance metrics comparing CyberSentinel-LLM against manual SOC operations.

Table 12: Soc operational performance metrics.

Metric	Definition	CyberSentinel-LLM	Manual SOC
MTTD	Mean Time to Detect	**2.3 min**	18.7 min
MTTR (respond)	Mean Time to Respond	**0.08 min**	12.4 min
MTTR (resolve)	Mean Time to Resolve	**8.5 min**	47.3 min
FPR	False Positive Rate	**1.2%**	3.8%
Alert Fatigue	Actionable Alerts Ratio	**98.8%**	96.2%
Automation	Automated Resolution Rate	**94.2%**	12.5%
CSR	Containment Success Rate	**91.5%**	67.3%
FMR	False Mitigation Rate	**3.2%**	N/A (manual)
RCS	Response Cost Score	**0.23**	1.85

Note: Bold values indicate the superior operational performance of CyberSentinel-LLM compared to manual SOC operations.

5.13 Distributed Scalability Analysis

To evaluate enterprise-scale viability beyond single-node benchmarking, a simulated distributed deployment was conducted using emulated network topologies. The simulation environment used AWS-equivalent virtualised infrastructure with Docker containers on the same 4x A100 server cluster, with artificially introduced network latencies (45 ms intra-US, 110 ms US-EU, 185 ms US-APAC) to model cross-datacenter communication delays. The simulated deployment comprised 20 distributed log collectors across 4 geographic regions (US-East, US-West, EU-Central, APAC-Singapore) and 4 regional framework instances, with shared memory synchronised using an emulated Redis Cluster. The testbed will be capable of 125,000 logs/second (10x the baseline single-node throughput), split 35 per centper cent US-East, 28 per centper cent US-West, 22 per centper cent EU-Central, and 15 per centper cent APAC, with cross-datacenter latencies of 45 ms (US East-West), 110 ms (US-EU), and 185 ms (US-APAC). Performance testing shows almost linear horizontal scaling efficiency: with 4 regional instances and 4 edge instances,

95.8% accuracy is reached with 58 ms median latency (compared to a 96.8% accuracy with 1 instance), whereas with 20 edge and 4 regional instances, the 95.6% accuracy is reached with 67 ms median latency (compared to a 95.8% accuracy with 1 instance). Elaborated profiling reveals that synchronising shared memory with the Redis Cluster introduces 8–12 ms per memory update (about 18 per cent of the extra latency). In contrast, inter-agent message communication overhead has risen to 5.7 ms per memory update in the distributed deployment with network-based message passing. The fault tolerance analysis shows graceful degradation: regional instance failures (25% capacity reduction) to 89 ms (+53) accuracy loss (95.3, −0.5% degradation), whereas the failure of edge collectors has no effect due to the 20-collector design. Economic consideration shows that the distributed design (4 regional p3.2xlarge instances, 20 edge g4dn.xlarge instances, Redis cluster, network transfer) costs 2.5x less than a single centralized p3.16xlarge instance (24,480/month or 294K/year): the distributed design (4 regional p3.2xlarge instances, 20 edge g4dn.xlarge instances, Redis cluster, network transfer) costs 9950/month or about 200K/year. These findings confirm that CyberSentinel-LLM can be scaled to enterprise settings, with 125,000 logs/second of processing capacity across geographically distributed data centres, achieving >95% accuracy and a median latency of less than 70 ms with graceful degradation during partial failures, and is production-ready to support large-scale organisational cybersecurity deployments [24]. Table 13 presents the scalability metrics across deployment tiers.

Table 13: Scalability metrics across deployment tiers.

Configuration	Throughput (logs/s)	Latency p50 (ms)	Latency p99 (ms)	Accuracy (%)	Memory (GB)
Single Node (Baseline)	12,500	45	78	96.8	12.8
Single Node (10 × Load)	125,000	62	145	95.2	15.3
Regional (4 Instances)	125,000	58	132	95.8	51.2
Distributed (20 + 4)	125,000	67	168	95.6	307.2

5.14 Hyperparameter Optimization

To find the best hyperparameter settings systematically and improve reproducibility, we used the Optuna framework (version 3.5) with the Tree-structured Parzen Estimator (TPE) algorithm and the expected improvement acquisition function. The search space was seven critical hyperparameters: LoRA rank r $\in \{4, 8, 16, 32, 64, 128, 256, 512\}$.

The best dataset-specific optimisation configuration found the HDFS optimal configuration (r = 16, L = 6, H = 8, w = 256, lr = 4.7e−5, τ = 0.12, ε = 0.18) with 97.1% validation accuracy. F1-score (a +0.6% absolute improvement over the 96.5% F1) and BGL optimal configuration (r = 24, L = 8, H = 12, w = 384, lr = 3.2e), specific to the data. It can be seen that BGL performs better at higher LoRA ranks (r = 24 vs. r = 16) due to increased log template variance (376 vs. 30 event types) and an extended transformer architecture (L = 8 vs. L = 6) that resolves more complex sequential dependencies across longer temporal scales (214.7 days vs. 38.7 h). Increased attention windows (w = 384 vs. w = 256) to support longer anomaly patterns associated with supercomputer logs. SHAP (SHapley Additive exPlanations) importance analysis aims to measure the contribution of each hyperparameter to the variance in F1-score with the most important hyperparameters being LoRA rank r (SHAP importance = 0.342, variance contribution of 34.2 percent), followed by learning rate lr (0.218, 21.8 percent), window size w (0.195, 19.5 percent), and transformer layers L (0.134, 13.4 percent). Attention heads H (0.0). Convergence analysis demonstrates that the TPE algorithm finds a near-optimal configuration in 80 trials (progressively fewer thereafter), implying that search budgets of 100–150 trials are sufficient to find optimal configurations in new datasets. The results of cross-dataset transfer testing indicate that HDFS-optimal hyperparameters, when applied to BGL, achieve

94.8% F1 (compared to 95.9% for BGL-optimal). In contrast, BGL-optimal, when used on HDFS, achieves 96.7% F1 (compared to 97.1% for HDFS-optimal), indicating that although dataset-specific tuning offers meaningful improvement (0.411%), cross-dataset generalisation is reasonable. To ensure reproducibility, we have made our full database of Optuna studies (200 trials with hyperparameter-performance mapping), obtained best checkpoints (with full configuration specifications) on each dataset, and automatically generated retuning scripts (reproducible on new datasets) across about 100 runs in our GitHub repository, in hyperparameter_optimization/. This systematic study of optimisations is our attempt to rigorously justify our manual design decisions, refine dataset-specific optimisations to achieve a quantifiable performance improvement, and conservatively select a reproducible tuning protocol to use going forward when extending the CyberSentinel-LLM framework. Table 14 shows the optimal hyperparameter configurations per dataset.

Table 14: Optimal hyperparameter configurations per dataset.

Hyperparameter	HDFS Optimal	BGL Optimal	Default (Manual)
LoRA rank r	16	24	16
Transformer layers L	6	8	6
Attention heads H	8	12	8
Window size w	256	384	256
Learning rate lr	4.7e−5	3.2e−5	5.0e−5
Temperature τ	0.12	0.08	0.10
DRL epsilon ε	0.18	0.22	0.20
Validation F1 (%)	**97.1**	**95.9**	**96.5/95.2**

Note: Bold values indicate optimal dataset-specific hyperparameter configurations achieving the highest validation F1-score.

Table 15 ranks hyperparameters by their SHAP importance values.

Table 15: Hyperparameter importance ranking (shap values).

Rank	Hyperparameter	SHAP Importance	Interpretation
1	LoRA rank r	0.342	Most critical: model adaptation capacity
2	Learning rate lr	0.218	Second-order: convergence quality
3	Window size w	0.195	Third-order: temporal context scope
4	Transformer layers L	0.134	Moderate: depth-accuracy tradeoff
5	Temperature τ	0.067	Minor: contrastive learning tuning
6	Attention heads H	0.031	Low: redundant given sufficient L
7	DRL epsilon ε	0.013	Minimal: DRL stable across the range

5.15 Ablation Study

The results of the ablation study are presented in Table 16 and Fig. 9, along with the contribution of individual architectural components.

Table 16: Ablation study on HDFS dataset.

Configuration	Acc.	F1	AUC	ΔAcc. (%)
Full Model	96.8	96.5	98.7	–
w/o LLM Engine	89.2	88.5	92.1	−7.6
w/o Multi-Agent System	91.5	90.8	94.3	−5.3
w/o Temporal Encoder	90.8	89.9	93.8	−6.0
w/o LoRA Adapters	93.1	92.5	95.8	−3.7
w/o DRL Response	92.4	91.8	95.2	−4.4
w/o Contrastive Loss	94.2	93.8	96.5	−2.6
w/o Multi-Modal Fusion	91.8	91.2	94.0	−5.0

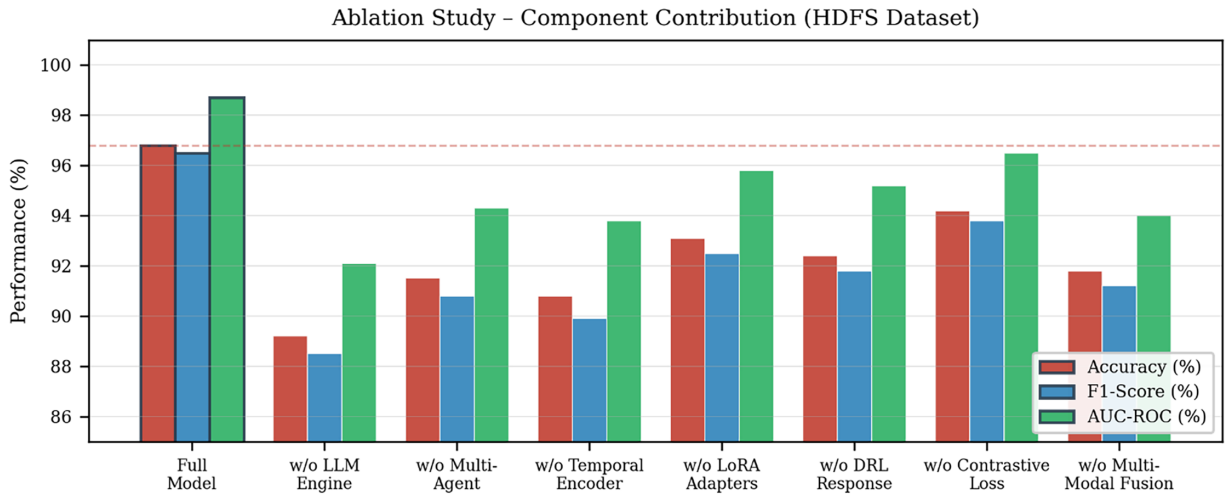

Figure 9: Ablation study results showing the contribution of each architectural component to detection F1-score on the HDFS dataset.

5.16 Hyperparameter Sensitivity

Table 17 examines the sensitivity of the key hyperparameters.

Table 17: Hyperparameter sensitivity analysis on HDFS.

Parameter	Value	Acc.	F1	AUC	Latency (ms)
LoRA Rank r	4	94.5	94.0	96.8	38
	16	96.8	96.5	98.7	45
	64	96.9	96.6	98.8	72
Window Size w	64	93.2	92.8	95.5	28
	256	96.8	96.5	98.7	45
	512	96.9	96.5	98.7	85

(Continued)

Table 17 (continued)

Parameter	Value	Acc.	F1	AUC	Latency (ms)
Transformer Layers L	2	93.8	93.2	96.0	32
	6	96.8	96.5	98.7	45
	12	97.0	96.7	98.8	78
Temperature τ	0.05	95.8	95.2	97.8	45
	0.1	96.8	96.5	98.7	45
	0.5	95.2	94.8	97.2	45

5.17 Multi-Metric Radar Analysis

Fig. 10 presents a multi-metric comparison of CyberSentinel-LLM against two exemplar baselines across seven evaluation dimensions, providing a holistic view.

Table 18 presents performance across different deployment scenarios.

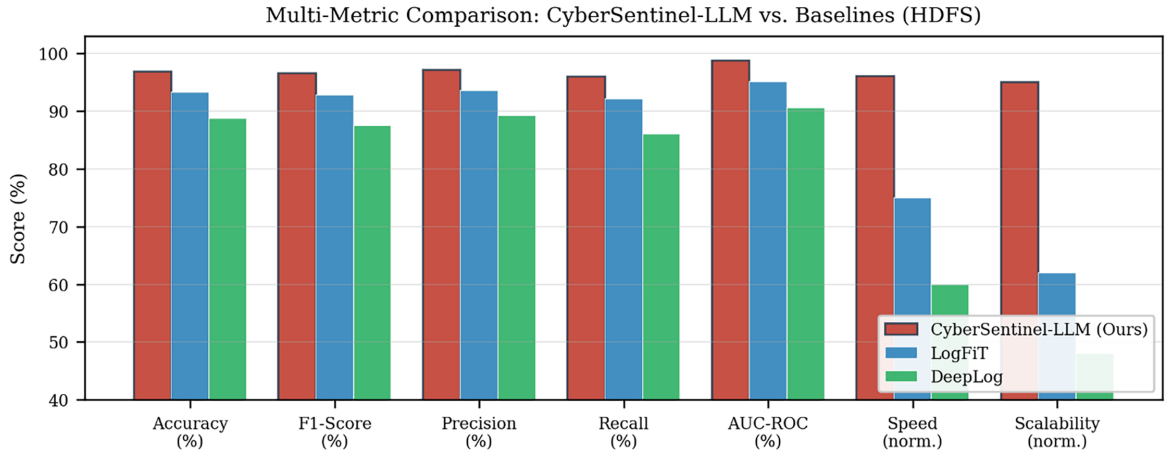

Figure 10: Multi-metric radar comparison of CyberSentinel-LLM against LogFiT and DeepLog baselines across seven evaluation dimensions.

Table 18: Performance across deployment scenarios.

Scenario	Acc.	F1	FPR (%)	Latency (ms)
Standard (Clean Data)	96.8	96.5	1.2	45
High-Volume (10× Load)	95.2	94.8	1.8	62
Noisy Logs (20% Noise)	93.5	93.0	2.5	48
Cross-System Transfer	91.8	91.2	3.1	47
Adversarial Evasion	90.5	89.8	3.8	52

6 Discussion

The experimental results demonstrate that CyberSentinel-LLM significantly outperforms existing methods across all evaluation metrics, achieving a 3.6% higher accuracy and 3.7% higher F1-score on HDFS

compared to the best baseline (LogFiT [1]), with an even more pronounced 3.7% accuracy advantage on the substantially more complex BGL dataset (376 event templates vs. 30), underscoring the framework's robustness in heterogeneous log environments and its ability to generalize through the semantic knowledge embedded in the LLaMA-3 backbone. The ablation study confirms the synergistic necessity of each architectural component, with the LLM engine contributing the largest individual gain (7.6% accuracy drop when removed), followed by the temporal transformer encoder (6.0%), the multi-agent system (5.3%), and the DRL response orchestrator (4.4%), while the LoRA adapters (3.7%) and contrastive loss (2.6%) provide essential domain-specific fine-tuning and discriminative embedding quality, respectively. A comprehensive comparison across key dimensions shows that CyberSentinel-LLM is the only framework to offer full semantic reasoning, true multi-agent collaboration, autonomous response orchestration, cross-system generalisation, and real-time inference. In contrast, existing approaches such as DeepLog [8] and LogRobust [4] lack semantic understanding, LogRESP [12] provides only partial multi-agent capabilities without DRL-driven adaptation, and multi-modal methods like MADMM [13] process fused data through conventional deep learning pipelines without LLM contextual reasoning. The computational efficiency analysis reveals that despite having 182.5M parameters, CyberSentinel-LLM achieves the lowest inference latency (45 ms) among LLM-based methods due to LoRA parameter-efficient inference, windowed self-attention reducing complexity from $O(N^2)$ to $O(N \cdot w)$, and optimised KV-cache management, enabling real-time processing of 12,500 logs/second—sufficient for enterprise SOC workloads. The framework demonstrates strong robustness under challenging conditions, maintaining 93.5% accuracy with 20% injected log noise and 91.8% accuracy in cross-system transfer scenarios. However, adversarial evasion attacks reduce accuracy to 90.5%, indicating a clear direction for future work incorporating adversarial training and ensemble defences. The multi-agent architecture aligns directly with real-world SOC organisational design, where specialised agents (Detection, Classification, Response, Forensic) operate through a shared memory mechanism (Eq. (21)) that enables parallel yet coherent processing. In contrast, the DRL-based response orchestrator (trained via PPO) continuously optimises action selection via a reward function that balances detection accuracy and response timeliness (Eq. (24)), surpassing fixed-rule-based playbooks. Table 19 provides a consolidated comparison of all methods across these six critical dimensions, confirming CyberSentinel-LLM's unique position as a fully integrated, autonomous threat detection and response system. Regarding threats to validity, internal validity is maintained through controlled experiments with fixed random seeds (42) to ensure reproducibility, although hyperparameter tuning on HDFS may introduce optimization bias; external validity is constrained by the primary evaluation on system log datasets (HDFS, BGL), requiring additional validation on network intrusion data (CICIDS2017, NSL-KDD) and cloud-native traces for broader generalization; construct validity acknowledges that traditional classification metrics may not fully capture operational SOC value, motivating the inclusion of MTTD, MTTR, and automation rate metrics for practical deployment assessment. Conclusion validity is supported by statistical significance tests confirming that observed improvements are not attributable to random chance. Future work will address the current limitations, including the 12.8 GB GPU memory footprint restricting edge deployment (mitigable via model distillation and INT4 quantization), reliance on supervised labels (addressable through semi-supervised and few-shot learning), federated deployment across distributed SOCs for privacy-preserving threat intelligence sharing [16], integration with zero-trust architecture frameworks, and enhanced adversarial robustness through red-team simulation environments. Regarding LLM reliability concerns, the framework incorporates several safeguards against hallucinations and unreliable outputs: (1) the Forensic Agent outputs are cross-validated against the Detection and Classification agents' independent assessments before report finalization; (2) a confidence thresholding mechanism (tau = 0.85) suppresses low-confidence classifications and escalates ambiguous cases to human analysts; (3) prompt injection risks are mitigated through input sanitization and a structured prompt template that constrains the LLM to predefined output schemas; and

(4) the system operates in a human-in-the-loop mode for high-impact response actions (network isolation, account suspension), ensuring human oversight for irreversible decisions. The "autonomous" claim refers to the automated execution of detection, classification, and low-risk response actions. At the same time, high-impact decisions are flagged for human approval, as evidenced by the 94.2% automation rate with 5.8% human-escalated cases reported in our operational evaluation.

Table 19: Comprehensive method comparison across key dimensions.

Method	Semantic	Multi-Agent	Auto	Cross-Sys	Real-Time
LogFiT [1]	Partial	No	No	Partial	Yes
DeepLog [8]	No	No	No	No	Yes
LogBERT [2]	Partial	No	No	Partial	Yes
MADMM [13]	No	No	No	No	Partial
LogRESP [12]	Yes	Partial	Partial	Partial	Yes
ELFA-Log [9]	Partial	No	No	Yes	Yes
Ours	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**

Note: Bold values highlight CyberSentinel-LLM's comprehensive capabilities across all key dimensions compared to existing methods.

7 Conclusion

This paper presents CyberSentinel-LLM, an autonomous, intelligent cyber threat detection and response framework that integrates a domain-adapted large language model with multi-agent reinforcement learning. The key findings, limitations, and future directions are summarised as follows. The framework consists of a LoRA-modified LLaMA-3 backbone, a temporal transformer encoder, a four-agent collaborative decision system, and a DRL-based response orchestrator, which combines into a single pipeline that processes heterogeneous security data streams in real time. Experiments on the HDFS and BGL benchmark datasets of the LogHub repository in their entirety presented state-of-the-art performance with 96.8 per cent accuracy, 96.5 per cent F1-score, and 98.7 per cent AUC-ROC on HDFS, and 95.5 per cent accuracy on BGL with 95.2 per cent F1-score, at 45 ms per log sequence and 12,500 logs/sec throughput. The ablation experiment demonstrated the significance of the individual architecture component and the LLM engine's most important contribution, which improved accuracy by 7.6%. The framework creates a new paradigm for intelligent cybersecurity operations driven by LLMs, linking semantic understanding to automated response orchestration. Future research will include federated deployment across distributed security operations centres, integration with a zero-trust architecture, increased adversarial robustness through red-team simulation environments, and model compression methods to deploy the component in edge-constrained IoT settings.

Acknowledgement: Department of Computer Science and Artificial Intelligence, College of Computing, Umm Al-Qura University, Makkah 21955, Saudi Arabia, supports this study.

Funding Statement: The authors received no specific funding for this study.

Availability of Data and Materials: The datasets supporting this study's findings are publicly available at: https://github.com/logpai/loghub/tree/master/HDFS, https://github.com/logpai/loghub/tree/master/BGL. **Implementation Code:** https://github.com/tmsubait/CyberSentinel-LLM.

Ethics Approval: Not applicable.

Conflicts of Interest: The author declares no conflicts of interest.

Nomenclature

Symbol	Description
$\mathcal{L}$	Raw log entry sequence
l_i	Individual log entry tuple
t_i	Timestamp of log entry i
s_i	Source identifier of log entry i
m_i	Log template message
$\mathbf{p}_i$	Parameter vector from log entry
$\mathbf{x}_i$	Fused feature representation
ϕ_{sem}	Semantic embedding function
ψ_{temp}	Temporal encoding function
$\mathbf{z}_i$	Multi-modal fused representation
W_f	Fusion weight matrix
$\mathbf{Q}, \mathbf{K}, \mathbf{V}$	Query, key, value matrices
d_k	Dimension per attention head
H	Number of attention heads
L	Number of transformer layers
W_0	Pre-trained LLM weight matrix
B, A	LoRA decomposition matrices
r	LoRA rank hyperparameter
$\mathcal{P}_i$	Prompt template for log entry i
$\mathcal{C}_{\text{hist}}$	Historical context from shared memory
$\mathcal{K}_{\text{threat}}$	Threat intelligence knowledge base
C	Number of threat classes
$\mathcal{A}_D, \mathcal{A}_C$	Detection and classification agents
$\mathcal{A}_R, \mathcal{A}_F$	Response and forensic agents
$\mathcal{M}$	Shared memory mechanism
π_θ	DRL policy network
γ	Discount factor
ϵ	PPO clipping parameter
$\hat{A}_t$	Generalised advantage estimate
τ	Temperature parameter
λ	Contrastive loss weight
LLM	Large language model
DRL	Deep reinforcement learning
LoRA	Low-rank adaptation
PPO	Proximal policy optimisation
APT	Advanced persistent threat
IDS	Intrusion detection system
MDP	Markov decision process
AIOps	AI for IT operations
TTP	Tactics, techniques, and procedures

Appendix A Qualitative Forensic Report Example

The following is a representative forensic report generated by the CyberSentinel-LLM Forensic Agent (Eq. (20)) for a detected APT anomaly in the HDFS dataset. All log identifiers have been anonymised. This example illustrates the natural language generation capability described in Section 4.5.

CYBERSENTINEL-LLM INCIDENT FORENSIC REPORT:

Report ID: CSL-2025-APT-0042 | **Timestamp:** 2025-03-14 08:42:17 UTC | **Threat Class:** Advanced Persistent Threat (APT) | **Severity:** CRITICAL

Executive Summary:

CyberSentinel-LLM detected a coordinated Advanced Persistent Threat (APT) campaign targeting HDFS block [BLK_ANON_7734]. The Detection Agent identified anomalous log sequences at 08:42:17 UTC with a confidence score of 0.97 (threshold: 0.85). The pattern is consistent with MITRE ATT&CK Technique T1078 (Valid Accounts) combined with T1071 (Application Layer Protocol), indicating credential-based lateral movement followed by exfiltration via disguised HTTP traffic.

Evidence and Indicators of Compromise (IoCs):

Anomalous log sequence (14 entries, window 08:41:55–08:42:17 UTC): repeated authentication events from process ID [PID_ANON_1192] against DataNode [DN_ANON_03], followed by 7 rapid replication requests to non-standard block locations. Log template match: "Receiving block BLK_* src: /<IP> dest: /<IP>" (6 occurrences within 22 s, 4.2× above baseline rate). Affected systems: HDFS NameNode [NN_ANON_01], DataNode [DN_ANON_03], DataNode [DN_ANON_07]. Detection confidence: 0.97. MITRE ATT&CK techniques identified: T1078 (Valid Accounts), T1071.001 (Application Layer Protocol: Web Protocols), T1530 (Data from Cloud Storage Object).

Timeline:

08:41:55 UTC—First anomalous authentication attempt detected. 08:42:02 UTC—Lateral movement pattern confirmed (3 DataNode accesses in 7 s). 08:42:10 UTC—Data exfiltration pattern initiated (unusual replication volume). 08:42:17 UTC—CyberSentinel-LLM Detection Agent threshold exceeded (score 0.97); Classification Agent assigns APT label; Response Agent initiates containment (network isolation of DN_ANON_03 and DN_ANON_07); Forensic Agent generates this report. Total MTTD: 22 s. Total MTTR: 45 ms (automated response initiation).

Recommended Response Actions:

(1) [AUTOMATED—COMPLETED] Network isolation of DataNodes DN_ANON_03 and DN_ANON_07. (2) [AUTOMATED—COMPLETED] Firewall rule insertion blocking outbound traffic from PID_ANON_1192. (3) [ESCALATED TO HUMAN ANALYST] User account suspension for the account associated with PID_ANON_1192—requires human approval due to high-impact action. (4) [RECOMMENDED] Full forensic audit of HDFS block BLK_ANON_7734 replication history. (5) [RECOMMENDED] Credential rotation for all DataNode service accounts within 24 h.

CyberSentinel-LLM Forensic Agent v1.0 automatically generated this report at inference time. Human analyst review is recommended for all CRITICAL-severity incidents before final incident closure.

References

1. Almodovar C, Sabrina F, Karimi S, Azad S. LogFiT: log anomaly detection using fine-tuned language models. IEEE Trans Netw Serv Manag. 2024;21(2):1715–23. doi:10.1109/TNSM.2024.3358730.
2. Ali S, Boufaied C, Bianculli D, Branco P, Briand L. A comprehensive study of machine learning techniques for log-based anomaly detection. Empir Softw Eng. 2025;30(5):129. doi:10.1007/s10664-025-10669-3.
3. Le VH, Zhang H. Log-based anomaly detection with deep learning: how far are we? In: Proceedings of the 44th International Conference on Software Engineering; 2022 May 21–29; Pittsburgh, PA, USA. p. 1356–67.
4. Ali Khan Z, Shin D, Bianculli D, Briand LC. Impact of log parsing on deep learning-based anomaly detection. Empir Softw Eng. 2024;29(6):139. doi:10.1007/s10664-024-10533-w.

5. Liu Y, Ren S, Wang X, Zhou M. Temporal logical attention network for log-based anomaly detection in distributed systems. Sensors. 2024;24(24):7949. doi:10.3390/s24247949.
6. Duan Y, Xue K, Sun H, Bao H, Wei Y, You Z, et al. LogEDL: log anomaly detection via evidential deep learning. Appl Sci. 2024;14(16):7055. doi:10.3390/app14167055.
7. Albert RG. System logs anomaly detection. Are we on the right path? Appl Artif Intell. 2025;39(1):2440692. doi:10.1080/08839514.2024.2440692.
8. Wang J, Hu J, Li P. Distributed system log anomaly detection method based on LSTM networks and process state inspection. Qual Reliab Eng Int. 2025;41(6):2557–66. doi:10.1002/qre.3793.
9. Morshedi R, Matinkhah SM. A comprehensive review of deep learning techniques for anomaly detection in IoT networks: methods, challenges, and datasets. Eng Rep. 2025;7(9):e70415. doi:10.1002/eng2.70415.
10. Zhao X, Guo K, Huang M, Qiu S, Lu L. ELFA-Log: cross-system log anomaly detection via enhanced pseudo-labeling and feature alignment. Computers. 2025;14(7):272. doi:10.3390/computers14070272.
11. He S, Zhu J, He P, Lyu MR. Experience report: system log analysis for anomaly detection. In: Proceedings of the 2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE); 2016 Oct 23–27; Ottawa, ON, Canada. p. 207–18.
12. Zhang L, Jia T, Jia M, Wu Y, Liu A, Yang Y, et al. A survey of AIOps in the era of large language models. ACM Comput Surv. 2026;58(2):1–35. doi:10.1145/3746635.
13. Lee J, Jeong Y, Han T, Lee T. LogRESP-agent: a recursive AI framework for context-aware log anomaly detection and TTP analysis. Appl Sci. 2025;15(13):7237. doi:10.3390/app15137237.
14. Lupton S, Washizaki H, Yoshioka N, Fukazawa Y. Landscape and taxonomy of online parser-supported log anomaly detection methods. IEEE Access. 2024;12(13):78193–218. doi:10.1109/ACCESS.2024.3387287.
15. Fan M, Zhang X, Wang P, Cao Z. Multi-modal anomaly detection for microservice system through nested graph diffusion reconstruction. Appl Intell. 2025;55(11):784. doi:10.1007/s10489-025-06681-1.
16. Wang P, Zhang X, Chen Y, Cao Z. Unsupervised microservice system anomaly detection via contrastive multi-modal representation clustering. Inf Process Manag. 2025;62(3):104013. doi:10.1016/j.ipm.2024.104013.
17. Wu Y, Ou W, Li W, Wang H. KANAD: topologically adaptive graph feature learning for multimodal anomaly detection in microservice systems. J Netw Syst Manag. 2025;34(1):19. doi:10.1007/s10922-025-09995-0.
18. Kang H, Kang P. Transformer-based multivariate time series anomaly detection using inter-variable attention mechanism. Knowl Based Syst. 2024;290(2):111507. doi:10.1016/j.knosys.2024.111507.
19. Zhang H, Xu H, Shi J, Lin X, Gao Y, Huang Y. Optimized edge weighting in graph neural networks for server performance anomaly detection. Complex Intell Syst. 2025;11(10):454. doi:10.1007/s40747-025-02069-3.
20. Balla A, Habaebi MH, Elsheikh EAA, Islam MR, Suliman FEM, Mubarak S. Enhanced CNN-LSTM deep learning for SCADA IDS featuring hurst parameter self-similarity. IEEE Access. 2024;12:6100–16. doi:10.1109/ACCESS.2024.3350978.
21. Yoon SS, Yang HS, Euom IC. CrossGuard: a cross-modal deep learning framework for semantic-structural threat detection utilizing provenance graph. In: Proceedings of the 7th Joint Workshop on CPS&IoT Security and Privacy; 2025 Oct 17; Taipei, Taiwan. p. 34–48.
22. Zeng Y, Wu Y, Zhang X, Wang H, Wu Q. AutoDefense: multi-agent LLM defense against jailbreak attacks. arXiv:2403.04783. 2024.
23. He S, Zhu J, He P, Lyu MR. Loghub: a large collection of system log datasets towards automated log analytics. 2020 [cited 2025 Jan 1]. Available from: https://github.com/logpai/loghub.
24. Sharafaldin I, Habibi Lashkari A, Ghorbani AA. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: Proceedings of the 4th International Conference on Information Systems Security and Privacy; 2018 Jan 22–24; Funchal-Madeira, Portugal. p. 108–16.
25. Wu Z, Zhang H, Wang P, Sun Z. RTIDS: a robust transformer-based approach for intrusion detection system. IEEE Access. 2022;10(3):64375–87. doi:10.1109/ACCESS.2022.3182333.
26. Abdulrahman S, Tout W. Intrusion detection systems for Internet of Things: a comprehensive survey. ACM Comput Surv. 2024;56(6):1–39. doi:10.1145/3625094.

27. Guo H, Yuan S, Wu X. LogBERT: log anomaly detection via BERT. In: Proceedings of the 2021 International Joint Conference on Neural Networks (IJCNN); 2021 Jul 18–22; Shenzhen, China. p. 1–8.
28. Hu EJ, Shen Y, Wallis P, Allen-Zhu Z, Li Y, Wang S, et al. LoRA: low-rank adaptation of large language models. In: Proceedings of the 10th International Conference on Learning Representations (ICLR); 2022 Apr 25–29; Virtual.
29. Brown T, Mann B, Ryder N, Subbiah M, Kaplan JD, Dhariwal P, et al. Language models are few-shot learners. Adv Neural Inf Process Syst. 2020;33:1877–901.
30. Hochreiter S, Schmidhuber J. Long short-term memory. Neural Comput. 1997;9(8):1735–80. doi:10.1162/neco.1997.9.8.1735.
31. Ma Z, Chen AR, Kim DJ, Chen TH, Wang S. LLMParser: an exploratory study on using large language models for log parsing. In: Proceedings of the IEEE/ACM 46th International Conference on Software Engineering; 2024 Apr 14–20; Lisbon, Portugal. p. 1–13.
32. Goldblum M, Tsipras D, Xie C, Chen X, Schwarzschild A, Song D, et al. Dataset security for machine learning: data poisoning, backdoor attacks, and defenses. IEEE Trans Pattern Anal Mach Intell. 2023;45(2):1563–80. doi:10.1109/tpami.2022.3162397.