LogRESP-Agent: A Recursive AI Framework for Context-Aware Log Anomaly Detection and TTP Analysis

JuYoung Lee , YeonSu Jeong , TaeHyun Han , TaeJin Lee *

Posted Date: 4 June 2025

doi: 10.20944/preprints202506.0235.v1

Keywords: endpoint log; log analysis; anomaly detection; semantic-aware; large language models (LLM); AI agents

 doi:10.20944/preprints202506.0235.v1

Article

LogRESP-Agent: A Recursive AI Framework for Context-Aware Log Anomaly Detection and TTP Analysis

JuYoung Lee †, YeonSu Jeong †, TaeHyun Han and TaeJin Lee *

Department of Information Security, Gachon University, Seongnam 13120, Republic of Korea
* Correspondence: kinjecs0@gmail.com
† These authors contributed equally to this work.

Abstract: As cyber threats become increasingly sophisticated, existing log-based anomaly detection models face critical limitations in adaptability, semantic interpretation, and operational automation. Traditional approaches based on CNNs, RNNs, and LSTMs struggle with inconsistent log formats and often lack interpretability. To address these challenges, we propose LogRESP-Agent, a modular AI framework built around a reasoning-based agent for log-driven security prediction and response. The architecture integrates three core capabilities, including (1) LLM-based anomaly detection with semantic explanation, (2) contextual threat reasoning via Retrieval-Augmented Generation (RAG), and (3) recursive investigation capabilities enabled by a planning-capable LLM agent. This architecture supports automated, multi-step analysis over heterogeneous logs without reliance on fixed templates. Experimental results validate the effectiveness of our approach on both binary and multi-class classification tasks. On the Monster-THC dataset, LogRESP-Agent achieved 99.97% accuracy and 97.00% F1-score, while also attaining 99.54% accuracy and 99.47% F1-score in multi-class classification using the EVTX-ATTACK-SAMPLES dataset. These results confirm the agent's ability to not only detect complex threats but also explain them in context, offering a scalable foundation for next-generation threat detection and response automation.

Keywords: endpoint log; log analysis; anomaly detection; semantic-aware; large language models (LLM); AI agents

1. Introduction

Endpoint logs are a critical source for monitoring system behavior and detecting malicious activity, capturing events such as process creation, user actions, and network connections [1,2]. As system environments become more complex, analyzing these logs plays a central role in security operations, including incident response, forensics, and threat hunting [3,4]. Traditional detection methods—ranging from rule-based systems to classical machine learning—have improved pattern recognition but remain limited in scalability, adaptability, and the ability to capture complex contextual or temporal relationships [5–8]. Deep learning models such as CNNs, RNNs, LSTMs, and Autoencoders further extended detection capabilities by modeling sequential patterns and latent structures [9–14]. However, these models often suffer from poor interpretability, require extensive feature engineering, and assume consistent log formats [15,16]. Recent research has explored Transformer-based models like LogBERT, which focus on sequence-aware modeling to improve anomaly detection [17]. While effective in structured environments, these models still face limitations in real-world deployment, especially in handling heterogeneous logs, explaining predictions, and supporting iterative reasoning [18,19]. More importantly, most existing models remain passive: they detect anomalies in isolated sequences but lack the ability to analyze contextual evidence, correlate across logs, or autonomously guide threat investigations [20,21]. This gap becomes critical in

 doi:10.20944/preprints202506.0235.v1

operational settings, where logs are noisy, diverse, and often require multi-step analysis to interpret complex attack behaviors.

In response to these challenges, we introduce LogRESP-Agent—a modular AI framework designed to:

1) Integrates LLM-based anomaly detection with semantic explanation to unify diverse log formats and support interpretable, context-aware threat analysis.
2) Employs Retrieval-Augmented Generation (RAG) and a suite of internal tools for recursive, multi-step investigation across heterogeneous logs.
3) Integrates a planning-capable LLM agent that generates human-readable explanations, allowing transparent and autonomous threat interpretation.

We evaluate LogRESP-Agent on the Monster-THC and EVTX-ATTACK-SAMPLES datasets. The agent achieves **99.97% accuracy and 97.00% F1-score** in binary classification, and **99.54% accuracy and 99.47% F1-score** in multi-class detection tasks. In addition to strong detection performance, the agent successfully generates context-aware explanations that align with the underlying threat behaviors. These results confirm that the framework not only excels in detection accuracy but also significantly improves interpretability—offering scalable support for automated security analysis.

1.1. Research Challenges

Analyzing real-world endpoint logs presents persistent challenges that limit the scalability, interpretability, and autonomy of current anomaly detection systems. We summarize the core problems motivating our work as follows:

1) Analyzing Heterogeneous and Unstructured Logs (Q1)

Real-world endpoint logs are generated from diverse systems, vendors, and security tools—resulting in inconsistent field names, data schemas, and levels of verbosity. This heterogeneity poses a significant challenge for models like LogBERT [17] and DeepLog [15], which rely on stable token vocabularies or template-based parsing. Almodovar et al. [19] demonstrate that such models are prone to semantic loss and generalization failure when applied to logs with unstructured formats or evolving schemas. Similarly, Zang et al. [23] point out that these approaches require retraining or manual normalization pipelines to maintain performance in multi-source environments, undermining their practical scalability.

2) Lack of Interpretability in Log Anomaly Detection (Q2)

Despite progress in anomaly detection, most existing models produce opaque outputs—such as binary labels or anomaly scores—without explanation. Ma et al. [20] found in a large-scale practitioner study that over 80% of analysts hesitated to act on unexplained alerts, citing a lack of trust and interpretability. Compounding this issue, Zamanzadeh et al. [21] further observe that most models lack temporal or contextual awareness, treating each log sequence in isolation and failing to correlate events across time or different sources. This hinders real-world triage and investigation workflows, where understanding *why* something is anomalous is as critical as the detection itself.

3) Limits of Static and Passive Inference Models (Q3)

Current AI-based detection models operate in a passive and static manner, performing inference in a single pass without external retrieval, multi-step reasoning, or goal-directed planning. As Yue et al. [23] note most LLM agents lack mechanisms to query external data or refine outputs during execution. Even frameworks like AutoGPT cannot verify plans or revise behavior mid-process, limiting their autonomy in investigative workflows [24]. Furthermore, Cemri et al. [25] report that multi-agent orchestration suffers from poor feedback control, making it difficult to scale analysis in dynamic environments requiring iterative, goal-driven reasoning.

1.2. Contributions

 doi:10.20944/preprints202506.0235.v1

To address these limitations, we propose LogRESP-Agent, a modular AI framework that combines LLM-based semantic analysis, context retrieval, and autonomous reasoning. The main contributions of this work are:

1. **Template-free Semantic Log Interpretation (C1)**

We design an LLM-based agent capable of directly analyzing unstructured and heterogeneous logs without relying on parsing templates. This enables broad adaptability across formats and mitigates schema drift, addressing *Q1*.

2. **Context-enriched Semantic Reasoning for Explanation (C2)**

Our framework includes a RAG module that retrieves relevant context—such as historical logs, process hierarchies, threat intelligence, and TTP knowledge—during inference. This allows the agent to enrich its analysis beyond isolated inputs and generate meaningful, explanation-driven outputs that align with real-world investigative needs, addressing Q2.

3. **Autonomous and Recursive Threat Investigation (C3)**

LogRESP-Agent is equipped with a planning-capable LLM that autonomously identifies missing context, sets investigative subgoals, and invokes internal tools—such as sequence scoring, TTP mapping, or process reconstruction—based on intermediate results. This enables recursive reasoning and dynamic adaptation during inference, supporting flexible, multi-step analysis workflows and addressing Q3.

2. Related Works

As log data becomes more complex and voluminous, a wide range of anomaly detection techniques have been proposed to support scalable and accurate analysis. These approaches can be broadly categorized into traditional machine learning models, sequence-based deep learning architectures, and LLM-driven autonomous agents.

2.1. Traditional and Deep Learning-Based Log Anomaly Detection

Early approaches to log anomaly detection primarily relied on rule-based systems and supervised machine learning models, such as Support Vector Machines (SVM), Random Forest (RF), and Decision Trees (DT), which used structured fields like timestamps, IP addresses, and port numbers as features [26–28]. These methods showed promising results when sufficient labeled data was available, and log structures remained consistent. However, they often struggled with generalization across heterogeneous environments and were brittle under unseen log formats [7,8].

To address the limitations of feature engineering and static modeling, researchers explored unsupervised and semi-supervised techniques—including k-means clustering, DBSCAN, Isolation Forest (IF) and One-Class SVM (OCSVM) —which required no labels and aimed to model normal behavior for outlier detection [29–32]. Among these, IF gained traction for its ability to detect rare events by recursively partitioning feature space.

As log data grew in complexity and volume, deep learning methods gained popularity for their ability to learn patterns directly from raw sequences.

Autoencoders were applied to learn compact representations and detect anomalies based on reconstruction errors [33].

Models like DeepLog [34] and LogAnomaly [35] leveraged LSTM-based sequence modeling to capture temporal patterns across log events, enabling improved detection of anomalous behavior in sequential logs. These models performed well on benchmark datasets but were often sensitive to minor log structure changes and lacked interpretability.

While these techniques improved detection performance, they lacked robustness to evolving log schemas and struggled with semantic understanding, especially when logs were unstructured or generated from varied sources such as Endpoint Detection and Response (EDR) agents or application-specific tools [19,22]. This ultimately motivated the shift toward transformer-based and LLM-enhanced approaches, which are discussed in the following section.

 doi:10.20944/preprints202506.0235.v1

2.2. Trnasformer-Based Log Anomaly Detection

Transformer architectures have significantly advanced log anomaly detection by capturing long-range dependencies and semantic relationships within log sequences—without the need for manual feature engineering. This has enabled more flexible modeling of log data compared to traditional sequence models.

Early works like HitAnomaly [36] used hierarchical encoding of templates and parameters to capture structure but were limited by their dependence on pre-parsed templates. NeuralLog [37] addressed this by embedding raw logs directly, increasing format robustness, though at the cost of reduced semantic depth due to shallow representations.

Later models such as LogBERT [17] and BERT-Log [38] applied masked prediction and fine-tuning to enhance sequence modeling. However, they often required retraining per dataset and struggled with unstructured or noisy logs. Generative models like LogGPT [18] and LogFiT [19] explored event prediction and masked sentence learning, improving adaptability but remaining limited in detecting subtle or multi-step anomalies.

These approaches share a fundamental limitation: they operate under a passive detection paradigm, processing logs as static sequences without the capacity for dynamic context integration or reasoning across distributed evidence. Overcoming these constraints requires a more proactive and adaptable framework—one that supports contextual reasoning, on-demand information retrieval, and iterative analysis in response to evolving threats. Effectively addressing real-world attack scenarios demands moving beyond static modeling toward intelligent systems capable of dynamic interpretation and multi-source correlation.

2.3. LLM Based AI Agent

The rapid advancement of Large Language Models (LLMs) has opened new avenues for intelligent log analysis. Unlike traditional methods, LLM-based agents can support contextual reasoning, tool coordination, and natural language explanation, offering the potential for more adaptive and interpretable security workflows. Recent research has leveraged these capabilities to automate tasks such as anomaly detection, threat triage, and incident response.

However, existing systems often rely on rigid, prompt-driven execution flows and lack the autonomy required for complex investigations. For example, IDS-Agent [39] combines LLM reasoning with external tools for intrusion detection, but its pipeline follows a fixed sequence and cannot revise its course mid-execution. Similarly, Security Event Response Copilot (SERC) [40] integrates Retrieval-Augmented Generation (RAG) into SIEM, enriching alerts with external intelligence—but only in response to predefined events, without proactive detection or recursive reasoning.

A more advanced design is found in Audit-LLM [41], which distributes analytical tasks across multiple agents (e.g., goal decomposers, script generators, executors). While this supports limited iteration, the overall flow remains pre-structured and agent behaviors are tightly scoped. These architectures cannot dynamically restructure goals, reselect tools, or adapt to ambiguous or evolving evidence without human intervention.

Overall, these approaches demonstrate the potential of LLM agents in automating security analysis, but also expose key limitations: most are constrained to single-pass logic, assume well-structured input, and lack self-directed reasoning. They are ill-suited to heterogeneous, semi-structured endpoint logs where threat indicators are fragmented or incomplete.

Addressing these gaps requires a new class of agents—capable of revisiting decisions, restructuring analytical goals, and coordinating tools in response to emerging context. In the following section, we introduce such a framework: an LLM-based agent architecture that supports recursive reasoning, dynamic planning, and self-guided anomaly investigation, enabling scalable and adaptable log analysis in real-world environments.

3. Proposed Method

 doi:10.20944/preprints202506.0235.v1

This section presents a three-stage framework for **log anomaly detection and autonomous threat analysis** using a self-directed AI agent. The architecture addresses the limitations of static models by supporting **goal-driven reasoning**, **tool coordination**, and **iterative analysis**.

As shown in **Figure 1**, the agent receives a user-defined objective (e.g., identify the attack type of a process) and operates through a plan–act–observe loop, selecting tools and refining its analysis as new evidence is gathered. To achieve this, the agent first interprets the given objective and selects relevant tools—such as a rule matcher, anomaly scorer, or threat mapper—based on the initial context. It then invokes these tools to extract meaningful signals from the log data, including anomaly scores, rule matches, and contextual history. The results are integrated to update the agent's internal hypothesis, and a natural language explanation is generated to summarize the findings. This loop continues adaptively until a confident conclusion is reached, enabling flexible and interpretable analysis of semi-structured logs. Each component is detailed in the following subsections.

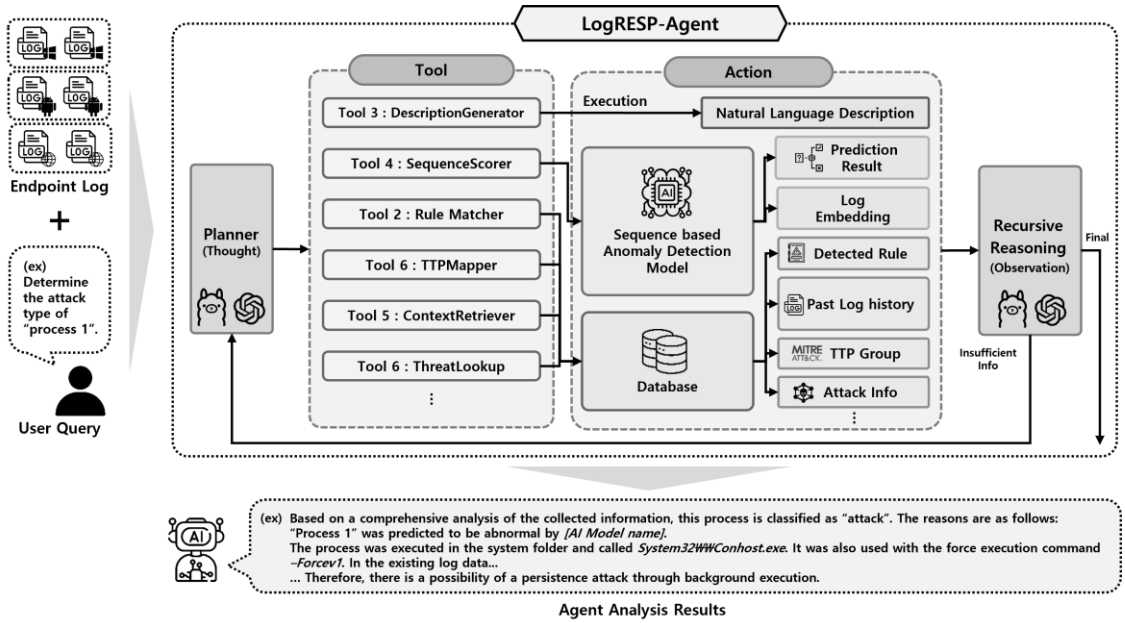

Figure 1. Overall architecture of the proposed LLM-based anomaly analysis framework, comprising three key stages: (1) Goal Planning — the agent interprets the user-defined objective and selects appropriate tools; (2) Dynamic Tool Execution — internal modules are invoked to extract anomaly scores, rule matches, and contextual information; and (3) Reasoning and Explanation — the agent synthesizes evidence, iteratively refines its hypothesis, and generates a natural language report.

3.1. Recursive Reasoning Framework for Log Analysis

To enable contextual and autonomous interpretation of log anomalies, we propose a recursive reasoning framework powered by a Large Language Model (LLM)-based agent. This framework is formally described in **Algorithm 1**, which outlines the core logic of the agent's recursive reasoning loop and illustrates how log analysis progresses through sequential planning, action, and evaluation steps. Unlike traditional pipelines that rely on static rules or single-pass inference, our framework employs a structured plan-act-observe loop that supports adaptive, multi-step reasoning. As illustrated in **Figure 2(b)**, the reasoning process is divided into three stages: Planning, Execution, and Reasoning.

Algorithm 1: Recursive Anomaly Analysis Loop

 doi:10.20944/preprints202506.0235.v1

Input: Semantic Log Description *D*, Analysis Goal *G*
Output: Final Explanation *E*

1. Initialize ObservationList ← [ ]
2. Initialize ThoughtHistory ← [ ]
3. While *True* do

Stage 1: Planning
a. Generate a new hypothesis h_t:
thought ← GenerateThought(D, G, ObservationList)
b. Select the most relevant tool:
tool ← SelectTool(thought)

Stage 2: Execution
c. Invoke the selected tool and get result:
result ← Invoke(tool, D)

Stage 3: Reasoning
d. Update observation and thought history:
ObservationList ← ObservationList ∪ {result}
ThoughtHistory ← ThoughtHistory ∪ {thought}
e. Determine if reasoning can be concluded:
if IsSufficient(result, ObservationList):
E ← GenerateExplanation(D, G, ObservationList)
break

4. Return *E*

1) Planning Strategy Formulation

The reasoning process begins with a user-defined or system-triggered analysis goal—such as *"Explain the anomaly detected at time TM"* or *"Determine whether process A is malicious."* Based on this goal, the agent retrieves the corresponding log record and generates a semantic description *D*, extracting key fields such as process name, parent lineage, path, privilege level, and execution context.

Based on this description, the agent formulates a hypothesis h_t about the nature of the observed behavior and selects the most relevant tool to investigate further. This step is guided by its internal knowledge of typical attack patterns and known threat signatures, which inform the selection of the most relevant investigative tool. Tool selection is guided by a utility function:

$$\text{tool}_t = \arg\max_{T_j \in \mathcal{T}} [\mu_1 \cdot \text{relevance}(T_j, G) + \mu_2 \cdot \text{expected\_gain}(T_j, O_{1:t-1})]$$

where $\mathcal{T}$ denotes the available toolset, G is the analysis goal, $O_{1:t-1}$ is the sequence of previous observations, and μ_1, μ_2 control the trade-off between goal alignment and anticipated information gain.

2) Context-Aware Execution

The selected tool is invoked to analyze the event or retrieve additional context. Tools may include *RuleMatcher*, *DescriptionGenerator*, *TTPMapper*, or *ContextRetriever*. The resulting insight r_t is stored in the agent's short-term memory, forming the basis for further reasoning. Detailed descriptions of each tool are provided in **Section 3.2**.

3) Reasoning via Recursive Threat Interpretation

In the final stage, the agent evaluates whether the current evidence is sufficient to draw a conclusion about the given analysis goal. If uncertainty remains, it returns to the Planning stage to reassess and continue the investigation. This forms a recursive loop of hypothesis generation, tool invocation, and result interpretation—formally described as the Thought–Action–Observation (TAO) cycle.

 doi:10.20944/preprints202506.0235.v1

To support this iterative reasoning, the agent maintains two memory structures. A short-term memory stores observations and tool outputs from the current session to inform immediate decisions. A global reasoning trace records the full sequence of thoughts, actions, and results, which are synthesized when generating the final explanation.

Reasoning termination is governed by a confidence-based decision policy:

$$terminate = \begin{cases} 1, if\ confidence_{malicious}(O) \geq \theta_1 \\ 1, if\ confidence_{benign}(O) \geq \theta_2 \\ 0, otherwise \end{cases}$$

The agent stops reasoning when there is sufficient evidence to support either a malicious or benign classification with high confidence.

This recursive reasoning framework supports several key capabilities:

- **Autonomous Planning**: The agent independently determines which tools to invoke and in what order.
- **Dynamic Context Expansion**: Relevant external information is retrieved as needed to fill knowledge gaps.
- **Recursive Hypothesis Refinement**: Each cycle incorporates new observations to update and improve its hypothesis
- **Goal-Directed Reasoning**: All steps are explicitly tied to the initial analysis objective, ensuring coherence

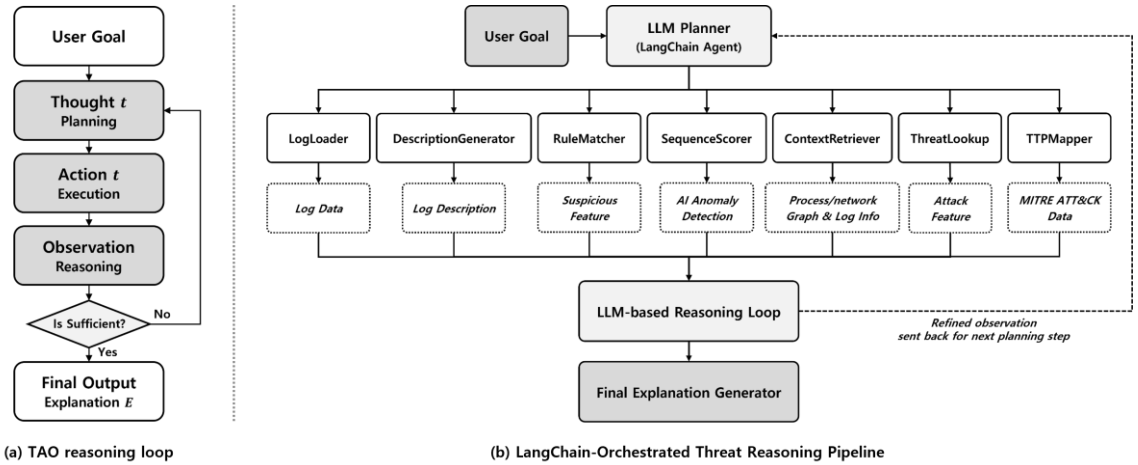

Figure 2. (a) TAO reasoning loop. Recursive process of hypothesis generation, tool invocation, and evidence evaluation executed by the LLM-based agent. (b) LangChain-Orchestrated Reasoning Pipeline. System-level architecture in which the LLM planner coordinates modular tools and integrates external context to iteratively construct threat explanations.

As illustrated in **Figure 2(a)**, the recursive TAO reasoning loop enables the agent to adaptively refine its analysis through iterative planning, execution, and reasoning cycles. This structure supports multi-hop inference across fragmented evidence and enables the generation of interpretable, goal-aligned explanations. Building on this, **Figure 2(b)** situates the reasoning process within a modular system architecture powered by a LangChain-based planner. This planner dynamically orchestrates specialized tools—such as *RuleMatcher*, *ContextRetriever*, and *TTPMapper*—that each contribute domain-specific insights. These components are not isolated utilities but serve as integral steps in the agent's evolving investigation, supporting tasks from rule-based matching to external context enrichment.

Together, the recursive reasoning loop and modular tool orchestration allow the system to emulate expert-level investigation strategies. By moving beyond static rule chains and embracing

 doi:10.20944/preprints202506.0235.v1

iterative hypothesis refinement, the proposed framework is well-suited for real-world security environments where log data is often noisy, incomplete, or ambiguous.

1.2. Tool-Oriented Semantic Transformation and Anomaly Detection

To support goal-directed log interpretation, the proposed agent integrates a suite of modular tools, each performing a distinct analytical role. These tools are not arranged in a fixed pipeline; instead, they are dynamically orchestrated by the LLM planner during each Thought–Action–Observation (TAO) cycle. At every reasoning step, the agent selects and invokes only the tools most relevant to the current hypothesis and available context. A summary of the tools and their corresponding functions is provided in **Table 1**.

Each invocation yields an observation appended to the agent's short-term memory and used in subsequent reasoning. Tools are revisited or bypassed adaptively: for example, the agent may switch from *RuleMatcher* to *ContextRetriever* when no known rule matches are found, then revisit the rule matching after additional context is retrieved.

This modular tool integration enables multi-perspective reasoning, where each tool contributes a complementary semantic or structural insight. Embedded within the recursive TAO framework, this architecture allows the agent to adaptively construct threat narratives, offering interpretable and context-rich explanations for both known and novel attack patterns.

Table 1. Summary of Tools Used for Semantic Transformation and Threat Inference.

Tool	Function
LogLoader	Structures raw log data from system or endpoint sources for further analysis
RuleMatcher	Applies rules to identify known malicious signatures
DescriptionGenerator	Converts structured logs into natural language event summaries
SequenceScorer	Scores log sequences based on behavioral anomalies using pretrained models
ContextRetriever	Gathers related process logs (e.g., parent/child) to support correlation
TTPMapper	Maps observed behaviors to MITRE ATT&CK TTPs
ThreatLookup	Fetches background information on threats or attacker tools from threat intel

3.3. Dynamic Analysis Cycle and Final Reasoning Output

Following iterative evidence collection via recursive reasoning, the agent proceeds to a final decision-making phase, where it synthesizes observations into a coherent explanation aligned with the original analysis goal. Unlike static anomaly detection systems that output binary labels or numerical scores, our agent adaptively revises its reasoning path based on evolving evidence and tool outputs.

Each TAO (Thought–Action–Observation) cycle is context-aware and hypothesis-driven. For example, if a rule match fails to yield results, the agent dynamically pivots to anomaly scoring or contextual retrieval. Conversely, the presence of multiple corroborating signals—such as elevated anomaly scores, matched TTPs, and relevant process ancestry—triggers early convergence toward a threat hypothesis.

Reasoning concludes when the agent determines that sufficient evidence has accumulated to justify a final decision. This decision is based on the semantic convergence of multiple signals, not a single heuristic.

To illustrate how the agent adapts to different investigation outcomes, Table 2 summarizes typical reasoning paths:

 doi:10.20944/preprints202506.0235.v1

Table 2. Reasoning Paths and Final Output Structure.

Scenario	Agent Behavior	Final Output
Attack	• Aggregate anomaly indicators • Cross-validate with TTP patterns • Confirm malicious intent	Identifies associated MITRE ATT&CK TTP(s) and describes supporting evidence in a structured explanation
Benign	• Review contextual consistency • Rule out attack hypotheses • Confirm benign justification	Explains why the event is benign (e.g., scheduled task, admin script), referencing safe patterns or known whitelist behaviors
Ambiguous /Low Confidence	• Detect insufficient or conflicting signals • Identify missing context • Defer or extend analysis	Continues reasoning or outputs "inconclusive" with explanation of missing context **o**r low-confidence factors

The final explanation output is designed to be human-interpretable and operationally useful, and typically contains:

(a) **Summary:** A high-level decision that characterizes the event as benign, suspicious, or indicative of an attack.

(b) **Evidence:** A focused set of key observations that played a central role in guiding the agent's assessment.

(c) **Reasoning Trace**: A chronological outline of the agent's investigative steps, detailing the sequence of tool invocations and the conclusions drawn at each stage.

(d) **Mapped Threat Context (if any)**: Behavioral correlations to known adversarial techniques, such as MITRE ATT&CK tactics, malware families, or threat actor patterns.

By generating natural language explanations grounded in both evidence and reasoning history, the system ensures not only interpretability and auditability but also operational usability. Each explanation captures what the agent observed, how it reasoned through multiple tools, and why it reached a particular conclusion—providing downstream analysts or automated response systems with the necessary context to validate, replicate, or act upon the results with confidence.

4. Implementation

To evaluate the practical utility of the proposed LangChain-based Security Agent, we conducted a series of experiments designed to assess both detection performance and explanation quality. Specifically, the evaluation focused on two key classification tasks: (1) Anomaly Detection of log entries as normal or malicious, and (2) Multi-class classification of malicious samples into specific attack types. In addition to performance metrics, we also analyzed the interpretability of the agent's outputs by examining its reasoning traces and final explanations. The following subsections detail the datasets used (4.1), overall implementation setup including baseline methods (4.2), detection results (4.3), and interpretability evaluation of the agent's reasoning process (4.4).

4.1. Datasets

We employed two datasets to support the anomaly detection and multi-class classification tasks of our experiments. The first dataset consists of endpoint process logs collected from a live enterprise environment, offering realistic benign and malicious activity patterns. The second dataset was curated from a publicly available repository of Windows attack logs, annotated with MITRE ATT&CK tactics. Each dataset was used for a distinct evaluation purpose.

1. **Monster-THC Endpoint Log**

This dataset was collected using the *Monster Agent*, a system-level event collector that serves as the endpoint component of the Monster Threat Hunting Cloud (THC) platform. While Monster

 doi:10.20944/preprints202506.0235.v1

Agent collects a wide range of system events—including process, network, and system-level activities—we selected process creation events (Event ID 1500) for this study. These logs correspond to execution activities of *Chrome*, *Edge*, and *Hwp* applications in a real-world Windows enterprise environment. The dataset contains 33,559 samples in total—33,272 labeled as benign and 287 as malicious—covering various attack types such as *Execution*, *Privilege Escalation*, and *Defense Evasion*. For the purposes of this study, we used the binary labels to evaluate the agent's performance in distinguishing normal and malicious behavior in real-world logs.

2. **EVTX-ATTACK-SAMPLES [42]**

The second dataset was constructed from the EVTX-ATTACK-SAMPLES [69] repository, which contains .evtx Windows logs corresponding to known adversarial behaviors. After converting the logs to structured CSV format, we used the provided MITRE ATT&CK tactic labels to annotate each sample. The dataset includes 3,167 malicious logs spanning eight attack categories—such as *Command and Control*, *Privilege Escalation*, and *Lateral Movement*—and was used to evaluate the agent's ability to classify specific attack types and generate appropriate explanations.

Table 3. Distribution of the Datasets Used for Anomaly Detection and Multi-Class Classification Tasks.

Dataset	Category (Process/Tactic)	Benign	Malicious	Total
Monster-THC	Chrome	30,036	150	30,186
	Edge	1,192	61	1,253
	Hwp	2,044	76	2,120
EVTX-ATTACK-SAMPLE	Command and Control	-	440	440
	Credential Access	-	218	218
	Defense Evasion	-	283	283
	Discovery	-	146	146
	Execution	-	381	381
	Lateral Movement	-	1,122	1,122
	Persistence	-	163	163
	Privilege Escalation	-	414	414

These datasets were used to evaluate the agent's classification performance and explanatory capabilities in separate binary and multi-class settings, as described in **Section 4.2.**

4.2. Experimental Configuration: Agent Components and Baselines

To evaluate both the detection performance and the interpretability of the proposed AI Agent, we conducted experiments on anomaly detection and multi-class attack classification. This section outlines the internal configuration of the agent and the baseline methods used for comparative evaluation.

1) AI Agent Architecture

The proposed agent is built on a modular reasoning framework coordinated by the Gemini-2.0-Flash large language model (LLM). The LLM orchestrates seven specialized tools to support semantic interpretation, anomaly scoring, rule-based matching, and threat mapping. Each tool is invoked dynamically based on the current reasoning context.

A detailed overview of these tools, including their roles, functions, and output formats, is provided in **Table 4** (see end of manuscript).

Unlike static pipelines, the agent performs recursive reasoning—selecting tools and interpreting results in context. Its flexible workflow enables multi-step analysis that adapts to intermediate findings.

Table 4. Tools Integrated in the AI Security Agent Architecture

 doi:10.20944/preprints202506.0235.v1

Tool Name	Role in Reasoning Loop	Function	Output Format
LogLoader	Data ingestion	Standardizes input logs into a unified format for downstream processing	Structured log in standard format
RuleMatcher	Signature-based detection (early stage)	Applies YARA/Sigma/custom rules to detect known suspicious patterns in process logs	Match result, rule metadata
Description Generator	Semantic summarization	Generates natural language summaries by pairing key log features with values to support semantic reasoning	Natural language sentence
SequenceScorer	Behavior modeling	Computes anomaly scores using LogBERT (anomaly detection) or classifies using RF/XGBoost (multi-class)	Score (0–1) or class label
ContextRetriever	Context expansion	Gathers related events (parent/child processes) to support behavioral correlation.	Related logs information (dict)
TTPMapper	Threat behavior mapping	Maps log behavior to MITRE ATT&CK techniques via cosine similarity between log descriptions and TTP embeddings (MiniLM-L12-v2).	Mapped TTP ID and label, Description
ThreatLookup	Intelligence enrichment	Provides concise descriptions of known TTPs and malware for enriched interpretation.	Textual threat summary

2) Baseline Methods

To ensure a comprehensive evaluation, we compared the AI Agent to three categories of baseline models:

(a) **Unsupervised anomaly detection models**. We included an Autoencoder and LogBERT as representative unsupervised methods. The Autoencoder detects anomalies based on reconstruction error, while LogBERT leverages masked language modeling to identify sequence-level deviations in system logs.

(b) **Supervised machine learning classifiers.** For multi-class classification, we trained standard classifiers including MLP, Random Forest, and XGBoost using structured log features. These models were selected for their widespread use in intrusion detection tasks and their strong performance on tabular data.

(c) **Agent-based ensemble variants.** In addition to standalone models, we evaluated two configurations of the proposed AI Agent: one using LogBERT for binary anomaly detection, and the others using either Random Forest or XGBoost for multi-class classification. These variants retain the same modular reasoning structure, with only the scoring component replaced.

By evaluating across these configurations, we aim to assess not only classification accuracy but also the interpretability and flexibility of the agent's recursive reasoning process.

4.3. Detection Performance Evaluation

1) Anomaly Detection Performance on Monster-THC Dataset

We evaluated the anomaly detection capabilities of LogRESP-Agent against two strong baselines—Autoencoder and LogBERT—across three process categories: Chrome, Edge, and Hwp. Evaluation metrics include True Positive Rate (TPR), False Positive Rate (FPR), Accuracy, and F1-score, where high TPR, Accuracy, and F1 combined with low FPR indicate robust detection performance.

As shown in **Table 5**, LogRESP-Agent consistently outperforms both baselines. Averaged across all processes, it achieved TPR 0.94, FPR 0.0, and F1-score 0.97—improving F1 by 15 percentage points over Autoencoder and 14 points over LogBERT. Notably, while LogBERT exhibited high accuracy, it frequently missed anomalies, as shown by its lower recall and F1.

Key findings include:

- **Chrome logs**: LogRESP-Agent achieved perfect detection (TPR = 1.0, F1 = 1.0) with zero false positives, whereas LogBERT and Autoencoder recorded significantly lower F1-scores (0.81 and 0.73, respectively).

 doi:10.20944/preprints202506.0235.v1

- **Edge logs**: The agent maintained strong performance (F1 = 0.94, TPR = 0.88), outperforming both baselines, which had comparable F1-scores (0.86) but lower recall.
- **Hwp**: On this more diverse process type, LogRESP-Agent again led with TPR 0.94 and F1-score 0.97, while LogBERT underperformed (TPR 0.68, F1 0.81), and Autoencoder plateaued at F1 0.86.

Table 5. Anomaly Detection performance of LogRESP-Agent compared with baseline models on the Monster-THC datasets.

Process	Autoencoder				LogBERT				LogRESP-Agent (Proposed)			
	TPR	FPR	Acc	F1	TPR	FPR	Acc	F1	TPR	FPR	Acc	F1
Chrome	0.79	0.002	0.99	0.73	0.71	0.0002	0.99	0.81	1.0	0.0	1.0	1.0
Edge	0.78	0.002	0.98	0.86	0.75	0.0	0.98	0.86	0.88	0.0	0.99	0.94
Hwp	0.78	0.002	0.99	0.86	0.68	0.0005	0.99	0.81	0.94	0.0	0.99	0.97
All	0.78	0.0002	0.98	0.82	0.71	0.0002	0.99	0.83	**0.94**	**0.0**	**0.99**	**0.97**

These results show that the agent not only inherits LogBERT's semantic understanding but further enhances detection accuracy through recursive reasoning and multi-tool integration. Tools like *RuleMatcher* and *DescriptionGenerator* contribute contextual insights that help reduce false alarms and detect subtle anomalies—boosting the system's practical utility in resource-constrained environments.

In summary, LogRESP-Agent offers consistent, interpretable, and highly reliable performance across all process types, confirming its effectiveness in real-world endpoint anomaly detection.

2) Multi-class Classification Performance on EVTX-ATTACK-SAMPLES

We further assessed LogRESP-Agent's ability to classify malicious logs into specific MITRE ATT&CK tactics using the EVTX-ATTACK-SAMPLES dataset. The agent was benchmarked against MLP, Random Forest (RF), and XGBoost, with performance evaluated across eight attack tactics.

As shown in **Table 6**, the XGBoost-based variant of LogRESP-Agent achieved the highest average TPR (0.99), FPR (0.001), Accuracy (0.99), and F1-score (0.99). Although the overall margin of improvement (~1%) over XGBoost may seem small, it reflects notable gains in harder classes like Credential Access and Privilege Escalation, where baselines showed degraded recall or increased false positives.

Highlights include:

- **Command and Control**: All models performed well (F1 ≥ 0.98), but only the agent variant achieved perfect detection (TPR = 1.0, F1 = 1.0) with zero false positives.
- **Credential Access**: A challenging tactic for baselines—MLP and RF scored F1 ≤ 0.87. The agent variants improved this to 0.96 (RF) and 0.97 (XGBoost) with very low FPRs (≤ 0.002).
- **Defense Evasion and Persistence**: Involving stealthy or multi-stage behaviors, these classes saw consistent improvements from the agent variants, reaching F1 = 0.97 while keeping FPRs ≤ 0.003.
- **Discovery and Execution**: Easier to detect, all models performed well, but LogRESP-Agent again maintained near-perfect scores (F1 ≥ 0.99, FPR = 0.0).
- In **Lateral Movement and Privilege Escalation**: While some baselines showed reduced F1 (0.87–0.90), LogRESP-Agent sustained F1 = 0.99–1.0 with minimal false positives.

Table 6. Multi-class Classification performance of LogRESP-Agent compared with baseline models on the EVTX-ATTACK-SAMPLES datasets.

Tactic	MLP				RF				XGBoost				LogRESP-Agent + RF (Proposed)				LogRESP-Agent + XGBoost (Proposed)			
	TPR	FPR	Acc	F1	TPR	FPR	Acc	F1	TPR	FPR	Acc	F1	TPR	FPR	Acc	F1	TPR	FPR	Acc	F1

 doi:10.20944/preprints202506.0235.v1

Command and Control	0.98	0.004	0.99	0.98	0.97	0.0	0.99	0.98	0.99	0.0	0.99	0.99	0.99	0.0	0.99	0.99	1.0	0.0	1.0	1.00
Credential Access	0.81	0.017	0.97	0.80	0.81	0.003	0.98	0.87	0.96	0.003	0.99	0.96	0.96	0.002	0.98	0.96	0.97	0.002	0.99	0.97
Defense Evasion	0.87	0.007	0.98	0.90	0.91	0.008	0.98	0.91	0.95	0.003	0.99	0.95	0.94	0.001	0.98	0.95	0.97	0.003	0.99	0.97
Discovery	0.93	0.0	0.99	0.96	0.96	0.0	0.99	0.98	1.0	0.0	1.0	1.0	0.99	0.0	0.99	0.99	1.0	0.0	1.0	1.0
Execution	0.97	0.0	0.99	0.98	0.96	0.0	0.99	0.97	0.98	0.002	0.99	0.98	0.99	0.0	0.99	0.99	0.99	0.0	0.99	0.99
Lateral Movement	0.96	0.02	0.97	0.96	0.98	0.01	0.98	0.97	0.99	0.005	0.99	0.99	0.99	0.004	0.99	0.99	0.99	0.001	0.99	0.99
Persistence	0.75	0.008	0.97	0.79	0.78	0.003	0.98	0.85	0.94	0.002	0.99	0.95	0.97	0.0	0.98	0.97	0.99	0.002	0.99	0.99
Privilege Escalation	0.95	0.02	0.97	0.90	0.95	0.03	0.96	0.87	1.0	0.002	0.99	0.99	0.98	0.008	0.98	0.98	1.0	0.0	1.0	1.0
All	0.9	0.01	0.98	0.91	0.92	0.007	0.98	0.93	0.98	0.002	0.99	0.98	**0.98**	**0.002**	**0.99**	**0.98**	**0.99**	**0.001**	**0.99**	**0.99**

By combining tree-based classifiers with recursive analysis, contextual reasoning, and TTP mapping, LogRESP-Agent delivers fine-grained and interpretable classifications across all tactics. The framework not only improves raw accuracy but also enhances tactical alignment—a critical requirement for operational threat analysis.

In conclusion, LogRESP-Agent extends beyond conventional classifiers by coupling high-fidelity detection with rich semantic explanations, making it well-suited for real-world, threat-informed defense and automated TTP attribution.

4.4. Explanation Analysis and Interpretability Evaluation

We also conduct a qualitative analysis to evaluate how well LogRESP-Agent explains its decisions. By reviewing three representative cases—one benign and one malicious log entry from the binary classification task, and one complex threat scenario from the multi-class classification setting—we assess the alignment between the agent's reasoning outputs and the actual attack context.

1) Case 1. Normal Chrome Execution (Monster-THC, Log ID: 1671)

In this scenario from the anomaly detection task, LogRESP-Agent analyzes a benign log entry corresponding to a Chrome process execution. The reasoning flow begins with the DescriptionGenerator, which produces a natural language summary of the event by extracting key attributes such as the parent process and command-line arguments. Rule-Matcher and SequenceScorer are then invoked, both returning benign outcomes—no known malicious patterns were found, and the log was evaluated as normal by the AI model.

To validate the context further, the agent employs ContextRetriever, which recon-structs the execution hierarchy and confirms that the Chrome process was launched by Slack—an expected behavior in enterprise environments. No anomalous flags or privilege escalation attempts were observed.

"No malicious patterns were detected. The execution chain matches typical Chrome browser behavior."

This case highlights LogRESP-Agent's ability to avoid false positives by combining rule-based detection, anomaly scoring, and contextual correlation. The resulting explanation is both accurate and interpretable, demonstrating the agent's effectiveness in identifying benign activity with high precision.

 doi:10.20944/preprints202506.0235.v1

Figure 3. Reasoning trace and final output of LogRESP-Agent for a benign Chrome log entry (#1671).

2) Case 2. Privilege Escalation via Chrome (Monster-THC, Log ID: 30048)

In this binary classification scenario, LogRESP-Agent analyzes a malicious log entry involving a multi-stage execution flow initiated from Chrome. The reasoning begins with the DescriptionGenerator, which summarizes the event involving conhost.exe and highlights suspicious command-line parameters. RuleMatcher immediately raises alerts for multiple known indicators: forced execution flags (e.g., *-ForceV1*), indirect command invocation (*cmd /c ComputerDefaults.exe*), and privilege-related tokens such as *SeChangeNotifyPrivilege*, all of which suggest stealth or traversal activity.

To further assess behavioral context, the agent invokes SequenceScorer, which classifies the log as abnormal, and ContextRetriever, which reconstructs a multi-process lineage. This reveals that Chrome was executed with sandbox restrictions disabled, launched PowerShell in hidden and non-interactive mode, downloaded a payload (msgbox.exe), which in turn executed cmd.exe with elevated commands. The sequence illustrates a staged privilege abuse attack across several process hops.

"Observed behavior includes indirect command execution, PowerShell in hidden mode, and multi-process escalation from Chrome to cmd.exe."

Finally, TTPMapper maps the observed pattern to multiple MITRE ATT&CK techniques, including T1202 (Indirect Command Execution) under the Defense Evasion tactic and T1548 (Abuse Elevation Control Mechanism) under Privilege Escalation. The agent's final malicious classification aligns precisely with the annotated ground truth and is well-supported by the combined results of all invoked tools.

This case demonstrates LogRESP-Agent's ability to surface latent threat signals by coordinating recursive reasoning steps. Its integration of rule-matching, anomaly scoring, and contextual reconstruction enables deep behavioral inference—moving beyond surface-level indicators to expose sophisticated attack strategies.

 doi:10.20944/preprints202506.0235.v1

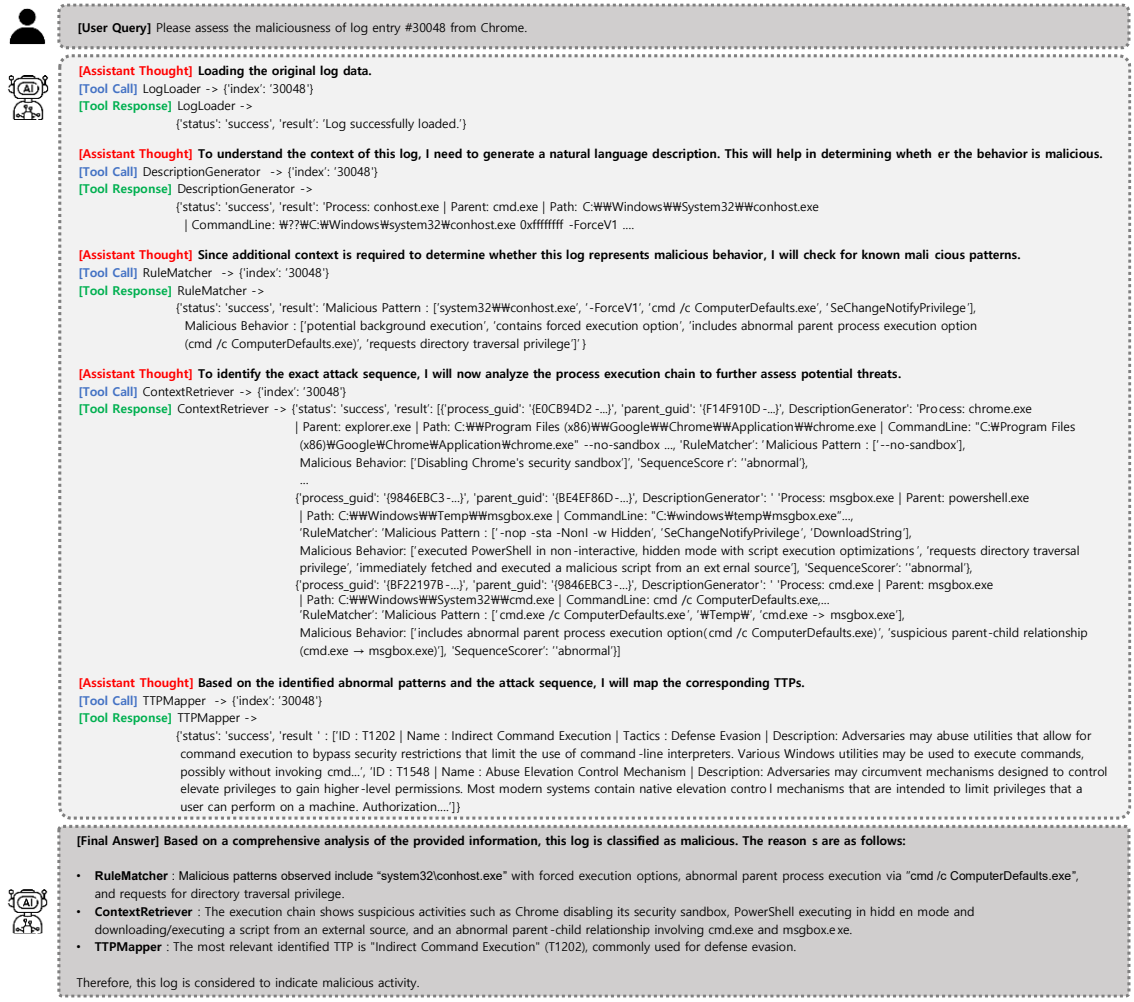

Figure 4. Reasoning trace and final output of LogRESP-Agent for attack Chrome log entry (#30048).

3) Case 3. Credential Access via IIS Tooling (EVTX, Log ID: 441)

The final case highlights LogRESP-Agent's reasoning in a multi-class classification scenario labeled as Credential Access. The event involves the execution of *appcmd.exe* from PowerShell, using suspicious arguments such as *list vdir /text:password*, which suggest attempts to enumerate credentials via IIS tooling. As the reasoning begins, the DescriptionGenerator captures this behavior in a natural language summary, emphasizing potential information extraction intent.

RuleMatcher then detects multiple known indicators of credential theft and evasion: use of obfuscation flags (*-nop -noni -enc*), execution from a *Temp* directory (a common sign of staging or evasion), and IIS-based credential access attempts via *appcmd.exe*.

To validate execution context, ContextRetriever reveals that the web server process *w3wp.exe* was the origin of the PowerShell instance, forming an abnormal parent-child hierarchy. This server-side compromise pattern deviates from normal usage and indicates possible remote exploitation.

"The process appcmd.exe was launched by PowerShell with arguments suggesting credential enumeration via IIS."

SequenceScorer, backed by a Random Forest classifier, assigns the final label "Credential Access." TTPMapper confirms this attribution by linking the behavior to MITRE ATT&CK techniques T1552.001 (Credentials in Files) and T1027 (Obfuscated Files or Information).

By combining rule-based matching, contextual reconstruction, and ML-driven classification, LogRESP-Agent delivers a precise and interpretable judgment. This case illustrates the agent's ability to detect layered credential access techniques with high semantic fidelity—transforming complex log evidence into clear tactical mappings for downstream analysts.

 doi:10.20944/preprints202506.0235.v1

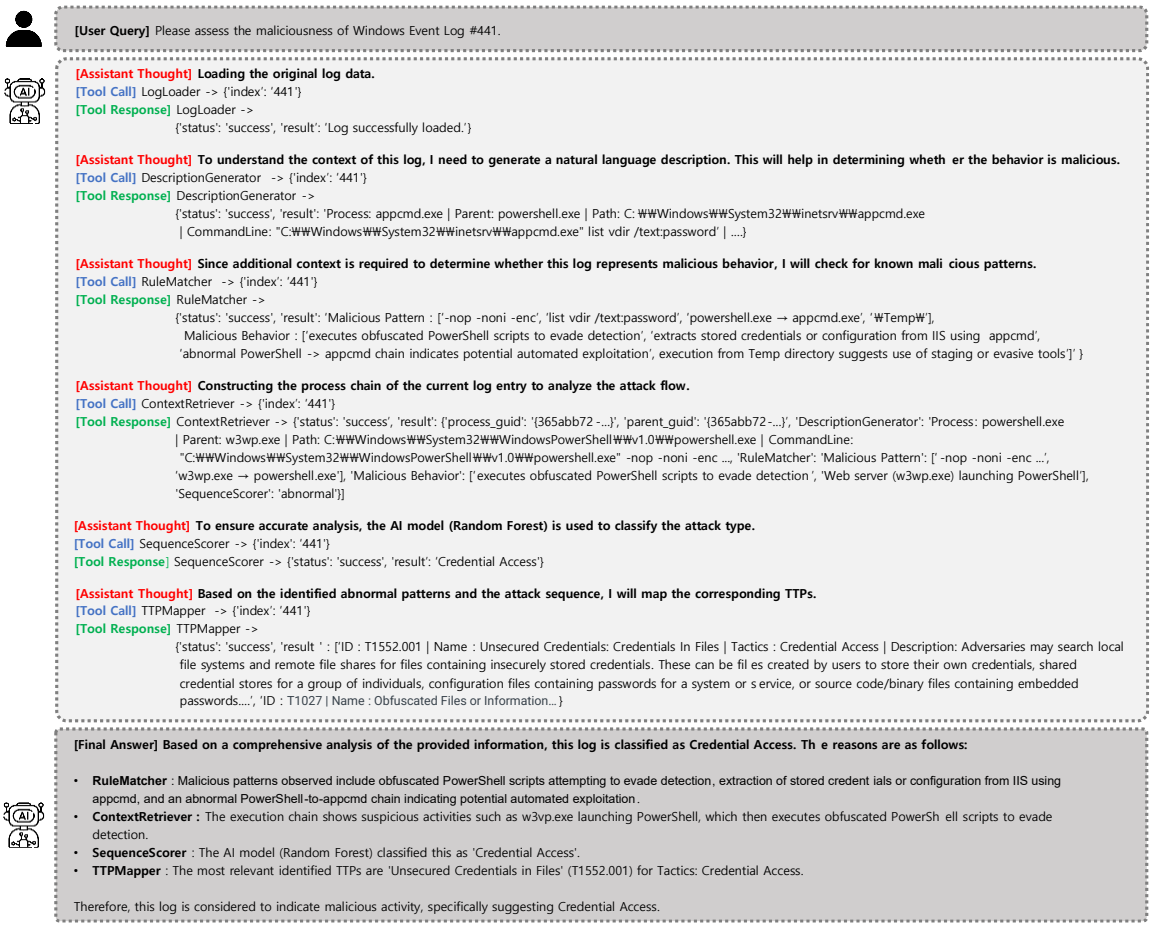

Figure 5. Reasoning trace and final output of LogRESP-Agent for attack Windows Event log entry (#441).

5. Discussion

This section discusses the practical implications, comparative advantages, and current limitations of the proposed LogRESP-Agent framework, as well as potential directions for future work.

1) Practical Value and Comparative Advantages

LogRESP-Agent offers high detection accuracy and interpretability without relying on structured input formats or fixed templates, making it well-suited for deployment in real-world, heterogeneous log environments. Its integration of semantic reasoning and TTP-based explanations enables Security Operations Centers (SOCs) to reduce triage time and improve incident response quality by delivering clear, actionable insights. In comparison to traditional models such as Autoencoders and LogBERT, LogRESP-Agent achieves superior performance while preserving transparency. Unlike static prompt-based agents (e.g., IDS-Agent or Wazuh-SERC), it utilizes a dynamic Thought–Action–Observation cycle that supports adaptive reasoning and context-aware decision-making, offering a more flexible and intelligent analysis capability.

2) Current

3) Limitations and Future Directions

Despite its strengths, LogRESP-Agent currently focuses on detection and explanation rather than direct automated response (e.g., generating firewall rules or isolating compromised hosts). Its accuracy and reliability are inherently dependent on the effectiveness of its component tools—such as RuleMatcher and SequenceScorer—and its robustness against sophisticated zero-day threats remains to be evaluated in adversarial environments. To address these limitations, future work will extend the framework toward proactive response capabilities, including mitigation recommendations like host isolation or process termination. We also plan to enhance cross-format

 doi:10.20944/preprints202506.0235.v1

and multilingual log processing to support broader operational environments. In addition, testing the agent's resilience against evasive attack strategies will be a priority, helping to advance LogRESP-Agent toward becoming a more autonomous and robust security co-pilot.

6. Conclusion

In this study, we proposed **LogRESP-Agent**, a modular AI framework that integrates log-based anomaly detection with contextual explanation capabilities. The system is driven by a high-performance Large Language Model (LLM), which dynamically orchestrates a suite of specialized tools—including pattern-based detection, sequence scoring, TTP mapping, and natural language generation—to support fine-grained threat classification and interpretable decision-making.

We evaluated LogRESP-Agent on both anomaly detection and multi-class classification tasks using the Monster-THC and EVTX-ATTACK-SAMPLES datasets. Across both settings, the proposed framework consistently outperformed conventional baselines—including standalone models (MLP, RF, XGBoost) and unsupervised methods (Autoencoder, LogBERT)—achieving up to 99.97% accuracy and 0.0 false positive rate in anomaly detection, and 0.99 F1-score in multi-class classification. It also maintained high true positive rates and robust performance across difficult attack categories such as Credential Access and Privilege Escalation.

Beyond detection performance, **Table 7** summarizes the broader system-level advantages of LogRESP-Agent over recent Transformer- and LLM-based anomaly detection models. Compared to models such as LogBERT, LogGPT, and Audit-LLM, LogRESP-Agent uniquely supports unstructured log reasoning, parser-free operation, autonomous goal-driven analysis, and dynamic multi-tool integration. Its recursive TAO reasoning loop, flexible tool extensibility, and output-level explainability establish it as a highly adaptive and analyst-friendly solution.

Table 7. Summary of model characteristics and limitations across Transformer-based, and LLM-based anomaly detection methods.

Model	Unstructured Log Support	Parser-Free Operation	Recursive Reasoning	Autonomous Analysis Flow	Goal Replanning	Multi-log Integration	Extensibility (Tool Integration + Automation)	Event-Level Reasoning	Inference Type	Output-Level Explainability
HitAnomaly [36]	×	×	×	×	×	×	×	△	Static	○
NeuralLog [37]	△	△	×	×	×	×	×	△	Static	△
LogBERT [17]	△	×	×	×	×	×	×	×	Static	△
BERT-Log [38]	×	×	×	×	×	×	×	△	Static	△
LogGPT [18]	×	×	×	×	×	×	×	△	Static	△
LogFiT [19]	○	○	×	×	×	×	×	△	Static	△
IDS-Agent [39]	△	△	×	△	×	△	△	△	Partially Dynamic	○
SERC [40]	×	△	×	△	×	△	○	△	Partially	○

 doi:10.20944/preprints202506.0235.v1

									Dynamic	
Audit-LLM [41]	△	△	×	△	×	△	△	○	Partially Dynamic	△
LogRESP-Agent (Proposed)	○	○	○	○	○	○	○	○	Dynamic	○

A key contribution of this work lies in demonstrating how recursive reasoning and modular tool orchestration can bridge the gap between static detection and real-world investigative workflows. By generating human-readable explanations aligned with MITRE ATT&CK and contextual process traces, LogRESP-Agent enhances both accuracy and actionability in modern SOC environments.

In future work, we plan to extend the agent's capabilities toward automated response generation, cross-format log interpretation, and resilience against adversarial threats. These directions aim to establish LogRESP-Agent as a scalable foundation for next-generation autonomous security analysis and decision support.

Author Contributions: Conceptualization, J.L. and Y.J.; methodology, J.L., Y.J. and T.H.; software, J.L. and Y.J.; validation, J.L., Y.J. and T.H.; formal analysis, J.L. and Y.J.; investigation, J.L. and Y.J.; resources, J.L., Y.J. and T.L.; data curation, J.L.; writing—original draft preparation, J.L. and Y.J.; writing—review and editing, J.L., Y.J., T.H. and T.L.; visualization, J.L.; supervision, T.L.; project administration, T.L.; funding acquisition, T.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded by the Institute of Information &Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) under Grant No. RS-2023-00235509, "Development of security monitoring technology based on network behavior against encrypted cyber threats in ICT convergence environment", and Grant No. RS-2024-00354169, "Technology Development of Threat Model/XAI-based Network Abnormality Detection, Response and Cyber Threat Prediction."

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: This study analyzed both public and private datasets. The EVTX-ATTACK-SAMPLES dataset is openly available on GitHub [42]. However, the Monster-THC dataset is not publicly available due to privacy and contractual restrictions, as it was provided by an industrial partner. Data sharing is subject to the provider's approval and cannot be disclosed without prior authorization. Requests to access this dataset should be directed to the corresponding author.

Acknowledgments: This work was supported by the Institute of Information &Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) under Grant No. RS-2023-00235509, "Development of security monitoring technology based on network behavior against encrypted cyber threats in ICT convergence environment" (70%), and Grant No. RS-2024-00354169, "Technology Development of Threat Model/XAI-based Network Abnormality Detection, Response and Cyber Threat Prediction." (30%)

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Mandru, S. K. Machine Learning and AI in Endpoint Security: Analyzing the use of AI and machine learning algorithms for anomaly detection and threat prediction in endpoint security. *Journal of Scientific and Engineering Research* **2021**, *8(3)*, 264-270.

 doi:10.20944/preprints202506.0235.v1

2. Karantzas, G.; & Patsakis, C. An empirical assessment of endpoint detection and response systems against advanced persistent threats attack vectors. *Journal of Cybersecurity and Privacy* **2021**, *1(3), 387-421.* [CrossRef]
3. Kara, I. Read the digital fingerprints: log analysis for digital forensics and security. *Computer Fraud & Security* **2021**, *2021*(7), 11-16. [CrossRef]
4. Smiliotopoulos, C.; Kambourakis, G.; Kolias, C. Detecting lateral movement: A systematic survey. *Heliyon* **2024**, *10*(4).
5. Lee, W.; Stolfo, S. J. A framework for constructing features and models for intrusion detection systems. *ACM transactions on Information and system security (TiSSEC)* **2000**, *3*(4), 227-261. [CrossRef]
6. Hofmeyr, S. A.; Forrest, S.; Somayaji, A. Intrusion detection using sequences of system calls. *Journal of computer security* **1998**, *6*(3), 151-180. [CrossRef]
7. Hussein, S. A.; Sándor, R. R. Anomaly detection in log files based on machine learning techniques. *Journal of Electrical Systems* **2024**, *20*(3s), 1299-1311.
8. Himler, P.; Landauer, M.; Skopik, F.; Wurzenberger, M. Anomaly detection in log-event sequences: A federated deep learning approach and open challenges. *Machine Learning with Applications* **2024**, *16*, 100554. [CrossRef]
9. Wang, Z.; Tian, J.; Fang, H.; Chen, L.; Qin, J. LightLog: A lightweight temporal convolutional network for log anomaly detection on the edge. *Computer Networks* **2022**, *203*, 108616. [CrossRef]
10. Lu, S.; Wei, X.; Li, Y.; Wang, L. Detecting anomaly in big data system logs using convolutional neural network. In 2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech), Athens, Greece, 12-15 August 2018 (pp. 151-158). IEEE. [CrossRef]
11. Meng, W.; Liu, Y.; Zhu, Y.; Zhang, S.; Pei, D.; Liu, Y.; etc. Loganomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs. In *IJCAI*, Macao, China, 10-16 August 2019 (Vol. 19, No. 7, pp. 4739-4745).
12. Zhang, X.; Xu, Y.; Lin, Q.; Qiao, B.; Zhang, H.; Dang, Y.; etc. Robust log-based anomaly detection on unstable log data. In Proceedings of the 2019 27th ACM joint meeting on European software engineering conference and symposium on the foundations of software engineering, Tallinn, Estonia, 26 – 30 August 2019 (pp. 807-817). [CrossRef]
13. Gu, S.; Chu, Y.; Zhang, W.; Liu, P.; Yin, Q.; Li, Q. Research on system log anomaly detection combining two-way slice GRU and GA-attention mechanism. In *2021 4th International Conference on Artificial Intelligence and Big Data (ICAIBD)*, Chengdu, China , 28-31 May 2021 (pp. 577-583). [CrossRef]
14. Farzad, A., & Gulliver, T. A. Unsupervised log message anomaly detection. *ICT Express* **2020**, *6*(3), 229-237. [CrossRef]
15. Du, M.; Li, F.; Zheng, G.; Srikumar, V. Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, Dallas, Texas, USA, 30 October 2017- 3 November 2017 (pp. 1285-1298). [CrossRef]
16. Wan, Y.; Liu, Y.; Wang, D.; Wen, Y. Glad-paw: Graph-based log anomaly detection by position aware weighted graph attention network. In *Pacific-asia conference on knowledge discovery and data mining*, Delhi, India , 11-14 May 2021 (pp. 66-77).
17. Guo, H.; Yuan, S.; Wu, X. Logbert: Log anomaly detection via bert. In *2021 international joint conference on neural networks (IJCNN)*, Shenzhen, China, 18-22 July 2021 (pp. 1-8). [CrossRef]
18. Han, X.; Yuan, S.; Trabelsi, M. Loggpt: Log anomaly detection via gpt. In *2023 IEEE International Conference on Big Data (BigData)*, Sorrento, Italy, 15-18 December 2023 (pp. 1117-1122). [CrossRef]
19. Almodovar, C.; Sabrina, F.; Karimi, S.; Azad, S. LogFiT: Log anomaly detection using fine-tuned language models. *IEEE Transactions on Network and Service Management* **2024**, *21*(2), 1715-1723. [CrossRef]
20. Ma, X.; Li, Y.; Keung, J.; Yu, X.; Zou, H.; Yang, Z.; etc. Practitioners' Expectations on Log Anomaly Detection. *arXiv preprint arXiv:2412.01066* **2024**.
21. Zamanzadeh Darban, Z.; Webb, G. I.; Pan, S.; Aggarwal, C.; Salehi, M. Deep learning for time series anomaly detection: A survey. *ACM Computing Surveys* **2024**, *57*(1), 1-42.

 doi:10.20944/preprints202506.0235.v1

22. Zang, R.; Guo, H.; Yang, J.; Liu, J.; Li, Z.; Zheng, T.; etc. MLAD: A Unified Model for Multi-system Log Anomaly Detection. *arXiv preprint arXiv:2401.07655* **2024**
23. Yue, M. A Survey of Large Language Model Agents for Question Answering. *arXiv preprint arXiv:2503.19213* **2025**.
24. Masterman, T.; Besen, S.; Sawtell, M.; Chao, A. The landscape of emerging ai agent architectures for reasoning, planning, and tool calling: A survey. *arXiv preprint arXiv:2404.11584* **2024**.
25. Cemri, M.; Pan, M. Z.; Yang, S.; Agrawal, L. A.; Chopra, B.; Tiwari, R.; etc. Why Do Multi-Agent LLM Systems Fail?. *arXiv preprint arXiv:2503.13657* **2025**.
26. Liang, Y.; Zhang, Y.; Xiong, H.; Sahoo, R. Failure prediction in ibm bluegene/l event logs. *Seventh IEEE International Conference on Data Mining (ICDM 2007)*. Omaha, Nebraska. 28-31 October 2007 (pp. 583-588). [CrossRef]
27. Wang, J.; Tang, Y.; He, S.; Zhao, C.; Sharma, P. K.; Alfarraj, O.; Tolba, A. LogEvent2vec: LogEvent-to-vector based anomaly detection for large-scale logs in internet of things. *Sensors* **2020**. *20*(9), 2451. [CrossRef]
28. Chen, M.; Zheng, A. X.; Lloyd, J.; Jordan, M. I.; Brewer, E. Failure diagnosis using decision trees. In *International Conference on Autonomic Computing,* New York, USA, 17-18 May 2004 (pp. 36-43) [CrossRef].
29. Dani, M. C.; Doreau, H.; Alt, S. K-means application for anomaly detection and log classification in hpc. In Advances in Artificial Intelligence: From Theory to Practice: 30th International Conference on Industrial Engineering and Other Applications of Applied Intelligent Systems, Arras, France, 27-30 June 2017 (Part 2, 30, pp. 201-210).
30. Mishra, A. K.; Bagla, P.; Sharma, R.; Pandey, N. K.; Tripathi, N. Anomaly Detection from Web Log Data Using Machine Learning Model. In *2023 7th International Conference on Computer Applications in Electrical Engineering-Recent Advances (CERA)*, Roorkee, India, 27-29 October 2023 (pp. 1-6). [CrossRef]
31. Karev, D.; McCubbin, C.; Vaulin, R. Cyber threat hunting through the use of an isolation forest. In *Proceedings of the 18th international conference on computer systems and technologies*, Ruse, Bulgaria, 23 – 24 June 2017 (pp. 163-170). [CrossRef]
32. Zhang, L.; Cushing, R.; de Laat, C.; Grosso, P. A real-time intrusion detection system based on OC-SVM for containerized applications. In *2021 IEEE 24th international conference on computational science and engineering (CSE)*, Shenyang, China, 20-22 October 2021 (pp. 138-145). [CrossRef]
33. Kramer, M. A. Nonlinear principal component analysis using autoassociative neural networks. *AIChE journal* **1991**, *37*(2), 233-243. [CrossRef]
34. Du, M.; Li, F.; Zheng, G.; Srikumar, V. Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, Dallas, Texas, USA, 30 October 2017- 3 November 2017 (pp. 1285-1298). [CrossRef]
35. Meng, W.; Liu, Y.; Zhu, Y.; Zhang, S.; Pei, D.; Liu, Y.; etc. Loganomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs. In *IJCAI*, Macao, China, 10-16 August 2019 (Vol. 19, No. 7, pp. 4739-4745).
36. Huang, S.; Liu, Y.; Fung, C.; He, R.; Zhao, Y.; Yang, H.; Luan, Z. Hitanomaly: Hierarchical transformers for anomaly detection in system log. *IEEE transactions on network and service management* **2020**, *17*(4), 2064-2076. [CrossRef]
37. Le, V. H.; Zhang, H. Log-based anomaly detection without log parsing. In *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, Melbourne, Australia, 15-19 November 2021 (pp. 492-504) [CrossRef]
38. Chen, S.; Liao, H. Bert-log: Anomaly detection for system logs based on pre-trained language model. *Applied Artificial Intelligence* **2022**, *36*(1), 2145642. [CrossRef]
39. Li, Y.; Xiang, Z.; Bastian, N. D.; Song, D.; Li, B. IDS-Agent: An LLM Agent for Explainable Intrusion Detection in IoT Networks. In *NeurIPS 2024 Workshop on Open-World Agents* **2024**.
40. Kurnia, R.; Widyatama, F.; Wibawa, I. M.; Brata, Z. A.; Nelistiani, G. A.; Kim, H. Enhancing Security Operations Center: Wazuh Security Event Response with Retrieval-Augmented-Generation-Driven Copilot. *Sensors (Basel, Switzerland)* **2025**, *25*(3), 870.

 doi:10.20944/preprints202506.0235.v1

41. Song, C.; Ma, L.; Zheng, J.; Liao, J.; Kuang, H.; Yang, L. Audit-LLM: Multi-Agent Collaboration for Log-based Insider Threat Detection. *arXiv preprint arXiv:2408.08902* **2024**.
42. **EVTX-ATTACK-SAMPLES.** Available online: https://github.com/sbousseaden/EVTX-ATTACK-SAMPLES (accessed on 30 May 2025).