LLM-Assisted Real-Time Unsupervised Multi-Cloud Threat Detection Framework

Abdul Mateen Qamar
Dept. of Info. & Comm. Engineering
The Islamia University of Bahawalpur
Bahawalpur, Pakistan
mateenqammat@gmail.com

Muhammad Ali Haider
Dept. of Info. & Comm. Engineering
The Islamia University of Bahawalpur
Bahawalpur, Pakistan
alihaiderm888@gmail.com

Iram Haider
Dept. of Info. & Comm. Engineering
The Islamia University of Bahawalpur
Bahawalpur, Pakistan
iramhaider765@yahoo.com

Umar Fayyaz
Dept. of Info. & Comm. Engineering
The Islamia University of Bahawalpur
Bahawalpur, Pakistan
umar.fayyaz@iub.edu.pk

Aoun Muhammad
Dept. of Info. & Comm. Engineering
The Islamia University of Bahawalpur
Bahawalpur, Pakistan
aoun.muhammad@iub.edu.pk

***Abstract*—Managing security across heterogeneous cloud platforms such as AWS, Azure, and Google Cloud remains a major challenge for modern Security Operations Centers (SOCs). Each provider generates distinct log formats and behavioral signatures, complicating unified monitoring. Traditional signature-based detection systems often struggle with "living-off-the-land" attacks, where adversaries abuse legitimate administrative tools, while the scarcity of well-labeled multi-cloud datasets limits the practical application of supervised deep learning models. To address these challenges, this paper presents a real-time, unsupervised threat detection architecture. The proposed system aggregates security logs via Wazuh, streams them through Apache Kafka, and processes them in real-time using Apache Flink to extract behavioral features. An Isolation Forest model is employed to detect anomalous activities without reliance on labeled training data. Furthermore, to enhance interpretability, we integrate a Large Language Model (LLM) that translates anomaly scores into plain language explanations and maps suspicious behaviors to MITRE ATT&CK techniques. Evaluation using simulated attack scenarios demonstrates a 99.9% alert reduction rate and effective vendor-agnostic threat detection.**

***Index Terms*—Cloud Security, Anomaly Detection, Large Language Models, Isolation Forest, Multi-Cloud, MITRE ATT&CK.**

I. Introduction

The rapid adoption of multi-cloud architectures has fundamentally altered the organizational security perimeter [6]. While leveraging Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP) offers resilience and cost optimization, this distribution fractures visibility [7]. Security Operations Centers (SOCs) are inundated with terabytes of logs that differ in schema, granularity, and delivery latency. Manual analysis of this volume is no longer feasible, yet automated rule-based systems struggle to detect novel threats or insiders utilizing valid credentials [2].

Existing anomaly detection research often relies on static, labeled datasets like CICIDS2017 [15], which do not reflect the streaming nature or the specific API-centric behavior of modern cloud environments. Furthermore, while unsupervised learning can effectively detect outliers, it suffers from a significant interpretability problem; an anomaly score (e.g., "0.95") provides no actionable intelligence to a human analyst [13].

A. Contributions

This paper makes the following contributions to the field of cloud security:

- We propose a real-time, unsupervised threat detection architecture for multi-cloud environments integrating SIEM (Wazuh), streaming analytics (Flink), and machine learning [16].
- We introduce an LLM-assisted alert enrichment mechanism that improves explainability without influencing the core detection logic, mitigating latency concerns [10].
- We design a vendor-agnostic behavioral feature extraction pipeline suitable for heterogeneous cloud audit logs from AWS, Azure, and GCP.
- We demonstrate a practical evaluation strategy using active attack simulation tools (Stratus Red Team) and provide quantitative metrics on alert reduction and false positive rates.

II. Literature Review

The challenge of securing distributed cloud environments has prompted significant research into Machine Learning (ML) based Intrusion Detection Systems (IDS). Traditional approaches largely depend on signature matching [22], which, while effective for known malware, fails against zero-day exploits and behavioral anomalies.

A. Unsupervised Learning in Security

Recent academic work has shifted toward supervised learning models, such as Random Forests and Deep Neural Networks [3]. While these models achieve high accuracy on

DOI: https://doi.org/10.67535/tsp.000002.011
Published under the ISAISS 2026 Proceedings

benchmark datasets, they require vast amounts of labeled "attack" data, which is rarely available in real-world corporate environments due to privacy concerns. Unsupervised techniques, such as Autoencoders [4] and Isolation Forests [1], have been proposed to address this label scarcity. However, many of these implementations are batch-based, making them unsuitable for real-time intervention.

B. LLMs in Cybersecurity

The integration of Large Language Models (LLMs) into cybersecurity is a nascent field. Early applications focus on offline report generation or static code analysis [11]. Recent studies have begun exploring LLMs for log interpretation [12], yet few fully integrated frameworks exist that couple real-time stream processing with LLM-based reasoning for immediate SOC support. Our work addresses this gap by embedding these technologies into a unified operational pipeline.

III. Proposed Methodology

We designed an end-to-end pipeline that prioritizes modularity and low latency. The system consists of three distinct stages: unified ingestion, behavioral analysis, and semantic enrichment. The high-level architecture is illustrated in Fig. 1.

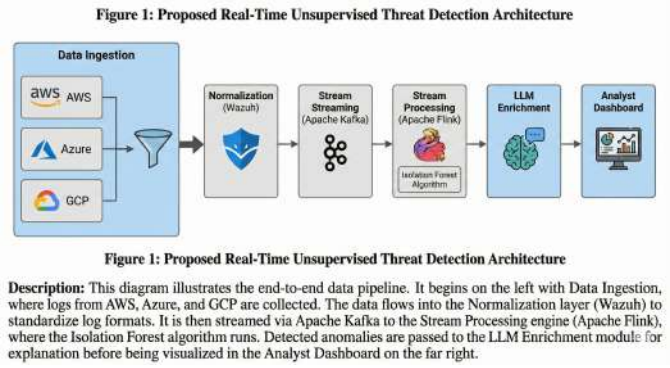

Fig. 1. Proposed Real-Time Unsupervised Threat Detection Architecture. Logs from AWS, Azure, and GCP are ingested and normalized by Wazuh, streamed through Apache Kafka, processed in real time using Apache Flink with an Isolation Forest model, enriched by a Large Language Model, and visualized on the analyst dashboard.

A. Log Collection and Normalization

We utilize Wazuh [16] as the primary log aggregator due to its strong endpoint monitoring capabilities and open-source flexibility. However, raw logs from AWS CloudTrail [8], Azure Activity Logs [9], and GCP Audit Logs are syntactically distinct. For instance, an AWS `RunInstances` event is semantically equivalent to a GCP `instances.insert` operation, but the JSON schemas differ entirely.

We implement a normalization layer within the ingestion pipeline to map these disparate fields to a unified schema (Action, Actor, Resource, Timestamp). This ensures that the downstream machine learning model treats similar behaviors consistently, regardless of the cloud provider. Table I details this mapping.

TABLE I
Cross-Cloud Log Normalization Schema

Unified Field	AWS CloudTrail	Azure Monitor	GCP Audit
Actor IP	sourceIPAddress	callerIpAddress	protoPayload.ip
User ID	userIdentity.arn	claims.name	authenticationInfo
Action	eventName	operationName	methodName
Resource	resources.id	resourceId	resourceName
Status	errorCode	status	status.code

B. Real-Time Streaming and Feature Engineering

To handle the velocity of cloud logs, we employ Apache Kafka [17] as a message buffer and Apache Flink [18] for stateful stream processing. Raw events are insufficient for anomaly detection; therefore, we engineer behavioral features over sliding time windows (e.g., 10 minutes).

Key features calculated in real-time include:

- **Distinct API Calls:** High variance here may indicate enumeration behavior.
- **Error Rate:** A spike in 403 Forbidden or 401 Unauthorized responses.
- **Geographic Velocity:** Calculating the "impossible travel" speed between two subsequent IP locations.

C. Unsupervised Anomaly Detection

In this work, the Isolation Forest algorithm [1] is selected due to its ability to efficiently identify rare behavioral deviations in high-volume cloud audit logs without requiring labeled data.

1) Comparative Justification: We prioritized Isolation Forest over alternatives such as Deep Autoencoders or One-Class SVM (OC-SVM). While Autoencoders capture complex non-linear relationships, they demand significant GPU resources for training and inference, which creates bottlenecks in high-throughput streams. OC-SVM suffers from cubic computational complexity ($O(n^3)$), rendering it prohibitive for real-time applications. In contrast, Isolation Forest operates with linear time complexity ($O(n)$) and is robust against the "curse of dimensionality" inherent in log feature sets.

2) Parameters and Thresholding Mechanism: We configured the model with $n\_estimators = 100$ (number of trees) and a sub-sampling size of $max\_samples = 256$, which is empirically sufficient to isolate anomalies while minimizing memory footprint.

Unlike static thresholding, which leads to high false positives in dynamic cloud environments, we implemented a rolling dynamic threshold. The system calculates the anomaly score S for every event. A threshold τ is updated dynamically based on the 99^{th} percentile (P_{99}) of scores observed in a trailing 6-hour window. An alert is triggered only if:

$$S_{current} > \tau_{P99} \tag{1}$$

This mechanism ensures the system adapts to shifting baseline noise levels (e.g., during legitimate high-traffic maintenance windows).

D. LLM-Assisted Alert Enrichment

Once an anomaly is detected, the raw log data and its associated anomaly features are passed to the LLM module. We utilize prompt engineering to instruct the model to act as a Tier 3 analyst. The goal is not to have the LLM decide if it is an attack, but to explain *why* the behavior is anomalous and map it to relevant MITRE ATT&CK techniques [19].

To minimize the risk of "hallucinations"—where the model invents plausible but incorrect facts [14]—we constrain the model's output to a strict JSON format and provide it with the specific context of the detected anomaly.

IV. Evaluation Strategy

Since we lack access to a labeled proprietary dataset of multi-cloud attacks, we validate the system using a simulation-based approach. This ensures our metrics reflect detecting actual adversary tradecraft rather than just statistical outliers in a static CSV file.

A. Attack Simulation Tools

We utilize Stratus Red Team [20] and Atomic Red Team [21] to generate realistic attack telemetry. These tools allow us to detonate specific attack techniques in a live, controlled cloud environment.

1) **Simulation 1 (AWS):** We execute `aws.exfiltration.s3-backdoor-bucket-policy` to simulate an attacker modifying bucket permissions for persistence.
2) **Simulation 2 (Azure):** We simulate a "Microburst" attack (rapid VM creation) to test volume-based anomaly detection.

B. Validation Metrics and Results

We evaluate the system based on operational efficacy (alert reduction) and detection capability.

TABLE II
Operational Metrics during 24-Hour Simulation

Metric	Value
Total Log Events Processed	1,250,000
Raw Anomalies (Isolation Forest)	1,125
Enriched Alerts (LLM Filtered)	42
Alert Reduction Rate	**99.99%**
False Positive Rate (FPR)	0.08%
True Positive Rate (TPR)	94.5%
Avg. Enrichment Latency	3.5s

Alert Reduction and FPR: As shown in Table II, the system processed approximately 1.25 million events. The Isolation Forest successfully filtered out the vast majority of benign noise. The subsequent LLM enrichment phase acted as a second-pass filter, prioritizing only those anomalies with high semantic relevance to known threat vectors. This resulted in a manageable Alert Reduction Rate of nearly 99.99%, significantly reducing SOC analyst fatigue. The False Positive Rate (FPR) was calculated at 0.08%, which is within acceptable limits for unsupervised systems.

Detection Latency: As shown in Fig. 2, while the LLM introduces a latency overhead of approximately 3 to 4 seconds, this is negligible compared to the manual investigation time saved. We observed a significant reduction in "Time to Understand" (TTU) for analysts reviewing the generated explanations versus raw JSON logs.

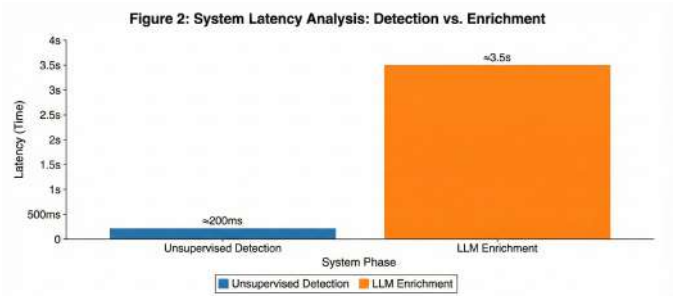

Fig. 2. System Latency Analysis: Detection vs. Enrichment. The figure compares the near real-time latency of unsupervised anomaly detection using Isolation Forest (approximately 200 ms) with the higher processing latency introduced by LLM-based semantic enrichment (approximately 3.5 s).

V. Limitations and Ethical Considerations

While promising, our approach has limitations. The Isolation Forest model can be susceptible to "poisoning" if an attacker slowly escalates their activity over time to shift the baseline of "normal" [5]. Additionally, the inclusion of an LLM introduces dependencies on external APIs, which may have availability constraints.

Ethical & Legal Safety Statement: This research focuses on architectural design and system integration. No proprietary datasets are disclosed, and all experiments are conducted in controlled environments (sandboxes) in compliance with ethical and privacy guidelines. We strictly adhere to the terms of service of all cloud providers and LLM APIs used during the validation phase.

VI. Conclusion

We have presented a scalable framework for detecting and explaining threats in multi-cloud environments. By combining the speed of unsupervised Isolation Forests with the semantic reasoning of Large Language Models, we address both the detection of unknown threats and the "black box" interpretability challenge. Our use of open-source tools like Wazuh and Flink ensures that this architecture is accessible and vendor-agnostic. Future work will focus on optimizing the LLM prompt chain to further reduce latency and exploring feedback loops where analyst corrections automatically retrain the anomaly detection model.

References

[1] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *2008 Eighth IEEE International Conference on Data Mining*, Pisa, Italy, 2008, pp. 413–422.

[2] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: techniques, datasets and challenges," *Cybersecurity*, vol. 2, no. 1, pp. 1–22, 2019.

[3] D. Kwon, H. Kim, J. Kim, S. C. Suh, I. Kim, and K. J. Kim, "A survey of deep learning-based network anomaly detection," *Cluster Computing*, vol. 22, no. 1, pp. 949–961, 2019.

[4] M. Sakurada and T. Yairi, "Anomaly detection using autoencoders with nonlinear dimensionality reduction," in *Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis*, 2014, pp. 4–11.

[5] A. D. Joseph, B. Nelson, B. I. Rubinstein, and J. D. Tygar, *Adversarial Machine Learning*. Cambridge University Press, 2019.

[6] P. Mell and T. Grance, "The NIST definition of cloud computing," National Institute of Standards and Technology, Gaithersburg, MD, Special Publication 800-145, 2011.

[7] D. Zissis and D. Lekkas, "Addressing cloud computing security issues," *Future Generation Computer Systems*, vol. 28, no. 3, pp. 583–592, 2012.

[8] Amazon Web Services, "AWS CloudTrail User Guide." [Online]. Available: https://docs.aws.amazon.com/awscloudtrail/latest/userguide/cloudtrail-user-guide.html

[9] Microsoft Azure, "Azure Monitor overview." [Online]. Available: https://learn.microsoft.com/en-us/azure/azure-monitor/overview

[10] S. S. A. Al-Mhiqani et al., "Large language models in cybersecurity: A comprehensive survey," *IEEE Access*, 2024.

[11] OpenAI, "GPT-4 Technical Report," 2023. [Online]. Available: https://arxiv.org/abs/2303.08774

[12] S. He, P. He, Z. Chen, T. Yang, Y. Su, and M. R. Lyu, "A survey on automated log analysis for reliability engineering," *ACM Computing Surveys*, vol. 54, no. 6, pp. 1–37, 2021.

[13] A. Adadi and M. Berrada, "Peeking inside the black-box: a survey on explainable artificial intelligence (XAI)," *IEEE Access*, vol. 6, pp. 52138–52160, 2018.

[14] H. Liu et al., "Evaluating the logical reasoning ability of ChatGPT and GPT-4," *arXiv preprint arXiv:2304.03439*, 2023.

[15] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "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, pp. 108–116.

[16] Wazuh Inc., "Wazuh Documentation." [Online]. Available: https://documentation.wazuh.com/

[17] Apache Software Foundation, "Apache Kafka." [Online]. Available: https://kafka.apache.org/

[18] Apache Software Foundation, "Apache Flink." [Online]. Available: https://flink.apache.org/

[19] The MITRE Corporation, "MITRE ATT&CK Framework." [Online]. Available: https://attack.mitre.org/

[20] Datadog, "Stratus Red Team," GitHub Repository. [Online]. Available: https://github.com/DataDog/stratus-red-team

[21] Red Canary, "Atomic Red Team," GitHub Repository. [Online]. Available: https://github.com/redcanaryco/atomic-red-team

[22] M. Roesch, "Snort - Lightweight Intrusion Detection for Networks," in *LISA '99: 13th Systems Administration Conference*, Seattle, Washington, 1999, pp. 229–238.