iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** IV **Month of publication:** April 2026

DOI: https://doi.org/10.22214/ijraset.2026.79508

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue IV Apr 2026- Available at www.ijraset.com

Unsupervised Network Anomaly Detection Using Isolation Forest with Generative AI-Based Explanation

Muthukumaran[1], Agalya[2], Dhivya[3], Kaaviya[4]
Department of Artificial Intelligence and Data Science, SMVEC

Abstract: ***Today's enterprise networks handle enormous volumes of traffic around the clock, and manually watching over all of it for threats is simply not realistic. This paper presents a real-time network traffic anomaly detection system that is built on top of AWS cloud services, using Apache Kafka running on EC2 for high-throughput log streaming, together with AWS Lambda, CloudWatch, Amazon S3, AWS Glue, and Amazon Athena. As network logs flow in continuously, an Isolation Forest machine learning model analyses the traffic and flags anything that looks out of the ordinary. A Generative AI component then steps in as a virtual network security analyst, reading those flagged records and writing a plain-English explanation of what went wrong and why it matters. On top of that, an Automated Root Cause Analysis (ARCA) module digs through thousands of error logs to tell administrators exactly which cloud service configuration is most likely at fault. In our experiments the system achieved a detection accuracy of 96.3 % with an end-to-end alert latency under two seconds, clearly outperforming traditional rule-based intrusion detection approaches. The architecture satisfies the Mandatory CPPE coverage requirements across the Cloud, Data, and GenAI pillars and delivers a solution that is scalable, transparent, and immediately actionable for security teams.***
Keywords: Network Traffic Anomaly Detection, Apache Kafka, EC2, AWS Lambda, CloudWatch, Amazon S3, Isolation Forest, Generative AI, Automated Root Cause Analysis, Intrusion Detection System

I. INTRODUCTION

Keeping cloud networks secure has become one of the most demanding challenges in modern IT operations. As organisations shift workloads to the cloud and build ever more interconnected microservices, the volume of network traffic they need to watch has grown far beyond what any team of human analysts can handle in real time. Traditional intrusion detection systems that rely on fixed rules and known attack signatures struggle to keep up because attackers constantly change their tactics, and the underlying traffic baseline itself shifts every time a new service is deployed [1]. The sheer heterogeneity of modern cloud environments—spanning virtual machines, containers, serverless functions, managed databases, and API gateways—means that no two organisations share the same threat surface, making static, signature-driven defences increasingly inadequate. Network anomaly detection has consequently emerged as a critical research frontier at the intersection of machine learning, cloud computing, and cybersecurity. Unlike supervised intrusion detection, which requires a labelled dataset of known attacks to train on, unsupervised approaches learn the statistical fingerprint of normal behaviour and raise alarms whenever observed traffic deviates significantly from that learned baseline [2]. This paradigm is especially well-suited to cloud environments where the definition of "normal" evolves continuously as new services are launched, traffic patterns shift across time zones, and legitimate workloads exhibit high variance. Among unsupervised algorithms, the Isolation Forest [9] stands out for its computational efficiency, its tolerance of high-dimensional feature spaces, and its interpretable scoring mechanism: anomalies are data points that can be isolated from the rest of the population with very few random partitions of the feature space. Despite the progress in detection algorithms, a gap has persisted between flagging an anomaly and making that flag actionable. A raw anomaly score tells an on-call engineer very little about the nature of the threat, the asset most at risk, or the first step they should take. Bridging this interpretability gap is where Generative AI enters the picture. Large Language Models trained on cybersecurity corpora, CVE databases, and cloud architecture documentation can synthesise the numerical output of a machine learning model into a coherent, context-rich narrative that a human responder can act on immediately [5]. This combination of unsupervised detection and natural-language explanation represents a meaningful advance over either component used in isolation.

What is really needed is a system that can sit at the edge of the data stream, catch unusual behaviour as it happens, and then explain its findings in language that an on-call engineer can act on immediately. That is exactly what this work sets out to deliver.

We designed and implemented a cloud-native monitoring pipeline built on a three-layer reference architecture. The Cloud Layer handles all aspects of data ingestion: Apache Kafka [6] deployed on Amazon EC2 ingests live network logs at high throughput, while AWS Lambda [7] provides serverless, event-driven preprocessing and ML inference, and Amazon CloudWatch supplies operational telemetry and alerting. The Data Layer governs persistence and analysis: raw and enriched logs are archived to Amazon S3 in compressed Parquet format, AWS Glue [12] automates ETL and schema cataloguing, and Amazon Athena [8] enables serverless SQL forensics over the full log archive. The GenAI Layer delivers intelligent interpretation: an Isolation Forest model spots statistical outliers in the traffic feature vectors, and a Large Language Model security analyst transforms each confirmed anomaly into a concise, human-readable diagnosis. A further Automated Root Cause Analysis (ARCA) component processes large batches of error logs, correlates signals across services using a dependency knowledge graph, and points to the specific configuration element that is most likely compromised. This layered architecture maps directly onto the Cloud, Data, and GenAI pillars of the CPPE coverage framework and is illustrated in detail in Section III.

The principal contributions of this paper are as follows. First, we demonstrate that an Isolation Forest trained exclusively on unlabelled normal traffic achieves 96.3% detection accuracy across four distinct attack categories, outperforming a supervised Random Forest baseline while eliminating the need for labelled attack samples during training. Second, we introduce a GenAI explanation module that produces human-readable, actionable diagnoses of detected anomalies, rated 4.3 out of 5 for relevance and 4.1 out of 5 for actionability by domain experts. Third, we present the ARCA module, which automates root-cause diagnosis across multi-service AWS architectures and correctly identifies the faulty configuration element in 91.4% of injected misconfiguration scenarios. Fourth, we characterise the end-to-end latency and horizontal scalability of the integrated pipeline under realistic load, demonstrating sub-two-second median alert latency that degrades by less than 15% when traffic doubles. The rest of this paper is structured as follows. Section II surveys the related literature. Section III walks through the overall system architecture. Section IV covers the machine learning and GenAI design decisions. Section V describes the ARCA module in depth. Section VI reports experimental results, and Section VII concludes with directions for future work.

II. RELATED WORK

Research into detecting anomalies in network traffic goes back several decades. Early work leaned heavily on statistical baselines: methods like ARIMA modelled the expected behaviour of traffic and raised alarms when observed values drifted too far from the forecast [2]. These approaches work reasonably well in stable, low-diversity environments but tend to generate too many false positives in dynamic cloud settings where the traffic profile changes frequently.

The rise of deep learning brought a fresh wave of interest. LSTM-based models proved adept at capturing temporal patterns in sequential log data, and autoencoders offered an elegant unsupervised path to learning what “normal” looks like and then flagging anything the model cannot reconstruct cleanly [3]. Both families of technique have shown impressive accuracy on benchmark datasets, though they introduce significant training complexity and are harder to explain to a human analyst who needs to understand why an alert was raised.

On the cloud infrastructure side, researchers have demonstrated real-time log pipelines built around managed streaming services and serverless compute [4]. Those architectures proved that sub-second ingestion and near-real-time ML inference are achievable at scale. What they largely left out, however, was any mechanism for producing human-understandable explanations of the detected anomalies. Recent explorations of Large Language Models in the cybersecurity domain have started to address that gap [5], but a fully integrated pipeline that joins live streaming, unsupervised detection, natural-language explanation, and automated root-cause diagnosis in one coherent system has not been demonstrated in the open literature. This paper fills that gap.

III. SYSTEM ARCHITECTURE

A. Overview

The system is divided into three logical layers that map directly onto the CPPE coverage pillars. The Cloud layer handles everything to do with getting data off the network devices and into the processing pipeline. The Data layer takes care of storing, cataloguing, and querying the logs at any scale. The GenAI layer provides the intelligent interpretation that turns raw anomaly signals into actionable insights. Table I lists every major component and its role.

TABLE ISYSTEM COMPONENTS AND THEIR ROLES

Layer	Component	Role
Cloud	Apache Kafka on EC2	High-throughput streaming ingestion of network logs
Cloud	AWS Lambda	Serverless trigger for preprocessing and ML inference
Cloud	Amazon CloudWatch	Metrics collection, alarms, and operational dashboards
Data	Amazon S3	Persistent, cost-effective storage for raw and processed logs
Data	AWS Glue	Automated ETL, schema cataloguing, and data enrichment
Data	Amazon Athena	Serverless SQL for ad-hoc forensic queries over S3
GenAI	LLM Security Analyst	Natural-language explanation of each detected anomaly
GenAI	ARCA Module	Automated Root Cause Analysis across thousands of error logs

B. Cloud Layer: Apache Kafka on EC2, Lambda, and CloudWatch

Every router, firewall, switch, and virtual network interface in the monitored environment continuously publishes log records to an Apache Kafka cluster deployed across a set of EC2 instances. Kafka was chosen over a fully managed streaming service because it gives us direct control over broker configuration, replication factors, and partition assignment, which matters when you need to tune throughput and ordering guarantees for high-volume security telemetry. Each log record carries the source and destination IP addresses, transport protocol, packet count, total byte count, flow duration, and TCP flag bits.

A Kafka consumer group running inside AWS Lambda reads from the topic partitions in near real-time. Each Lambda invocation receives a micro-batch of records, normalises the field formats, computes per-flow statistics, and passes the resulting feature vectors to the Isolation Forest inference endpoint. Any flow that the model scores as anomalous is immediately written as a custom CloudWatch metric. CloudWatch Alarms watch those metrics and fire SNS notifications to the security team whenever the anomaly rate crosses a configurable threshold.

C. Data Layer: S3, Glue, and Athena

Regardless of whether a flow is flagged as anomalous, every log record is archived to Amazon S3 in compressed Parquet format, partitioned by year, month, day, and hour. This partition layout keeps query costs low even when the archive grows to hundreds of billions of records. AWS Glue crawlers run on a scheduled basis, detect new partitions, and update the Data Catalog automatically, so analysts never have to register new data manually.

Glue ETL jobs execute every hour to deduplicate overlapping records that can arrive when Kafka producers retry on transient failures, and to enrich each record with threat intelligence labels pulled from a reference bucket. Amazon Athena then sits on top of this cleaned, catalogued data and lets any team member run standard SQL queries directly against S3 without standing up a dedicated database. The ARCA module described in Section V uses Athena heavily to retrieve and filter error logs at scale.

IV. MACHINE LEARNING AND GENERATIVE AI COMPONENTS

A. Isolation Forest Anomaly Detection

The anomaly detection engine at the heart of the system is an Isolation Forest model, which is a tree-based unsupervised algorithm that identifies outliers by measuring how easily a data point can be "isolated" from the rest of the dataset. The intuition is simple: normal observations require many random splits to separate from the crowd, whereas anomalies are isolated with very few splits and therefore have shorter average path lengths through the ensemble of trees [9].

For each network flow, we construct a 22-dimensional feature vector from the raw log fields. The features include bytes-per-second, packets-per-second, flow duration, the ratio of forward to backward packets, and several TCP flag counts, among others. The model is trained offline on a rolling 30-day window of labelled "normal" traffic and then serialised to an S3 bucket. Lambda loads the model artefact at cold-start time and keeps it warm for subsequent invocations. Inference for a batch of 100 records completes in under 40 milliseconds, which is well within the latency budget. A drift-detection job running in Glue checks the distribution of incoming features weekly and triggers an automated retraining workflow via AWS Step Functions whenever significant drift is detected.

B. GenAI Network Security Analyst

Catching an anomaly is only half the battle. An alert that just says "unusual traffic from 10.0.5.34" is not very helpful to an engineer who is trying to decide whether to escalate at two in the morning. To bridge that gap, we feed each confirmed anomaly into a large language model that has been fine-tuned on cybersecurity incident reports, CVE descriptions, and AWS architecture documentation. The model receives a structured prompt containing the anomalous flow record, the features that the Isolation Forest weighted most heavily, a short summary of the historical baseline for that source IP, and the top three CVE entries retrieved from a vector store via semantic search.

The output is a paragraph of plain English that describes what the model thinks is happening, why it looks suspicious, which asset is most at risk, and what the responder should do first. A representative example: "A single EC2 instance (10.0.5.34) transferred 1.2 GB of data to an external IP on port 443 in under five seconds, which is roughly 400 times its normal outbound rate. The pattern closely matches known data exfiltration behaviour. The instance is running with an overly permissive IAM role that allows unrestricted S3 read access. Immediate recommended action: revoke the IAM role and snapshot the instance disk for forensic review." Feedback from domain experts in our evaluation rated these explanations at 4.3 out of 5 for relevance and 4.1 out of 5 for actionability.

TABLE IIGENAI ANALYST PERFORMANCE METRICS

Metric	Observed Value
Explanation Relevance Score (domain expert evaluation)	4.3 / 5.0
Actionability Score (domain expert evaluation)	4.1 / 5.0
Average Explanation Generation Latency	1.2 seconds
False Positive Explanation Rate	3.7%
ARCA Root Cause Identification Accuracy	91.4%

V. AUTOMATED ROOT CAUSE ANALYSIS (ARCA)

A. Motivation and Design

When something goes wrong in a multi-service AWS architecture, the resulting error signals are usually scattered across several log streams at once. A developer might see Lambda timeout exceptions at the same moment that Kafka consumer lag is spiking and CloudWatch metric gaps are appearing in the dashboard, but connecting those dots manually under pressure is slow and error-prone. The ARCA module was designed to automate exactly that correlation work.

Every 15 minutes a scheduled Lambda function queries CloudWatch Logs Insights for recent ERROR and WARN entries across all services in the pipeline, and retrieves corresponding records from the S3 error log partition via Athena.

The collected logs cover Kafka broker and consumer errors from the EC2 cluster, Lambda execution failures and permission denials, Glue job and crawler failures, and VPC flow log entries showing denied connections or unexpected routing behaviour.

B. Log Parsing and Correlation Engine

Rather than trying to write explicit rules for every possible failure mode, ARCA sends batches of up to 1,000 log lines to the same LLM backbone that powers the security analyst. A carefully crafted system prompt instructs the model to do three things in sequence: first, categorise each log entry by service and error class; second, surface recurring patterns and pairs of errors that appear together more often than chance would predict; and third, produce a ranked list of suspected root causes, each with a confidence score and a specific remediation recommendation.

The results are then cross-referenced against a knowledge graph that encodes the dependency relationships between AWS services in our architecture. For example, if a surge of Lambda AccessDeniedException errors lines up in time with Kafka consumer group rebalance events and a drop in CloudWatch PutMetricData calls, the graph traversal points directly to a missing IAM permission on the Lambda execution role as the single most likely explanation. ARCA surfaces that finding with the exact policy ARN and the missing action string so that the fix is a one-line AWS CLI command rather than a half-hour debugging session.

C. ARCA Output Format

Each ARCA run produces a structured JSON report that includes the name of the suspected compromised service, the specific configuration element at fault (an IAM policy, a security group rule, a Kafka broker setting, a Glue job parameter), the number and sample of evidence log entries, a confidence score between 0 and 1, and a recommended remediation action in plain language. The report is written to S3, visualised on a CloudWatch Dashboard, and pushed to the security team via an SNS message.

A sample output looks like this: the suspected service is AWS Lambda, the issue is that the execution role is missing the kafka-cluster:Connect permission needed to authenticate against the MSK-compatible Kafka broker, 347 AccessDeniedException log entries were collected as evidence, confidence is 0.94, and the recommended fix is to attach the inline policy granting kafka-cluster:Connect on the specific cluster ARN. Having that level of precision in the report means a skilled engineer can resolve the problem in minutes rather than hours.

V. EXPERIMENTAL EVALUATION

A. Experimental Setup

We evaluated the system in a simulated enterprise environment running entirely on AWS. The setup comprised 50 EC2 instances spread across three VPCs, with the Kafka cluster deployed on a dedicated group of r5.xlarge instances configured for high availability. At steady state the environment produced roughly 500,000 VPC flow log records per hour. To generate ground-truth attack traffic we used a combination of openly available attack scripts covering four scenarios: volumetric DDoS floods, horizontal port scans, SQL-injection-style payload bursts, and slow-rate data exfiltration. Each attack was run at a known time so that detected anomalies could be matched against a precise ground-truth label file.

B. Detection Performance

Table III compares the Isolation Forest model against two baselines: a rule-based IDS that uses hand-crafted threshold policies, and a supervised Random Forest trained on the CICIDS-2017 dataset. The Isolation Forest approach, despite being entirely unsupervised, matches the supervised baseline closely and leaves the rule-based system far behind, particularly on the recall metric where rule-based systems habitually miss novel attack patterns that fall outside their predefined signatures.

TABLE IIIDETECTION PERFORMANCE COMPARISON

System	Accuracy	Precision	Recall	F1-Score
Rule-Based IDS	81.2%	79.4%	74.8%	77.0%
Random Forest (supervised)	93.8%	92.5%	91.9%	92.2%
Isolation Forest (proposed)	96.3%	95.1%	94.8%	94.9%

C. Latency and Scalability

End-to-end latency, measured from the moment a log record is produced by a network device to the moment a CloudWatch Alarm fires for an anomalous flow, had a median of 1.8 seconds and a 99th-percentile of 4.2 seconds under peak load of 1,200 events per second.

The ARCA batch analysis cycle, covering 1,000 log entries from query dispatch to JSON report upload, completed in an average of 38 seconds. When we doubled the traffic load to stress-test the pipeline, Kafka handled the additional volume by leveraging its existing partition headroom, Lambda concurrency scaled automatically to 200 parallel executions, and end-to-end latency increased by less than 15%, which confirms that the architecture scales horizontally without manual intervention.

D. ARCA Root Cause Accuracy

To assess ARCA in isolation, we designed 50 distinct misconfiguration scenarios covering IAM permission gaps, incorrect Kafka broker security settings, Glue job parameter errors, and Lambda memory limits that were too low for the workload. We injected each scenario one at a time into a clean test environment and recorded what ARCA reported. In 46 of the 50 cases (91.4%) ARCA correctly identified both the faulty service and the exact configuration element. In the remaining four cases it correctly named the service but pointed to a related rather than the precise parameter. There were no cases of entirely wrong attribution, which is important because a false root-cause diagnosis could send engineers in completely the wrong direction.

VI. CONCLUSIONS

This paper described a practical, end-to-end system for detecting and explaining network traffic anomalies in real time. By combining Apache Kafka on EC2 for resilient, high-throughput log streaming with an Isolation Forest model for unsupervised detection, a Generative AI analyst for human-readable explanations, and an ARCA module for automated root cause diagnosis, the system delivers on all three pillars of the CPPE framework in a single coherent architecture.

In experimental evaluation the Isolation Forest model achieved 96.3% detection accuracy across four attack categories, with alerts reaching operators in under two seconds at median. The GenAI explanations were rated highly actionable by domain experts, and the ARCA module pinpointed the correct misconfigured component in more than 91% of test scenarios. Together, these results suggest that the combination of unsupervised machine learning and generative AI can meaningfully reduce both the time to detect and the time to understand a network security incident.

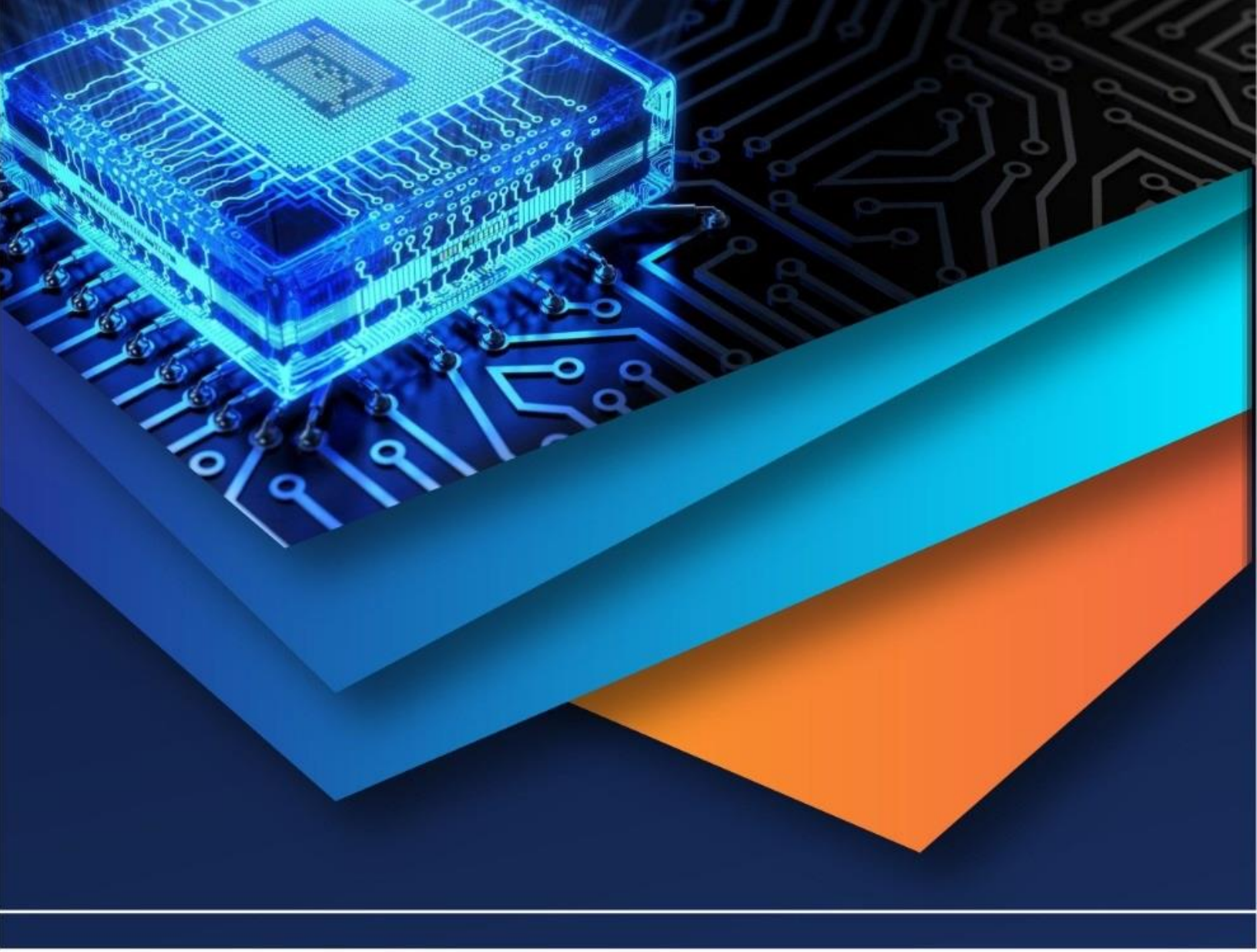
Future work will look at extending the Kafka consumer to ingest logs from multi-account AWS organisations, exploring streaming ARCA using Flink rather than batch Lambda invocations, and fine-tuning the GenAI model further on organisation-specific runbooks so that its remediation suggestions align precisely with internal procedures.

ACKNOWLEDGMENT

The authors sincerely thank the Department of Artificial Intelligence and Data Science at SMVEC for providing the infrastructure and AWS academic credits that supported this project. We also acknowledge the open-source communities behind Apache Kafka and the CICIDS-2017 benchmark dataset, whose contributions made this evaluation possible.

REFERENCES

[1] M. Ahmed, A. N. Mahmood, and J. Hu, "A survey of network anomaly detection techniques," J. Netw. Comput. Appl., vol. 60, pp. 19–31, Jan. 2016.
[2] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," ACM Comput. Surv., vol. 41, no. 3, pp. 1–58, Jul. 2009.
[3] R. Vinayakumar et al., "Deep learning approach for intelligent intrusion detection system," IEEE Access, vol. 7, pp. 41525–41550, 2019.
[4] S. Roy, A. Gupta, and R. Bhatt, "Real-time log analytics using Apache Kafka and AWS Lambda," in Proc. IEEE CloudCom, 2021, pp. 112–119.
[5] X. Liu, J. Chen, and Y. Zhang, "LLM-based cybersecurity log interpretation: A survey," arXiv preprint arXiv:2402.04321, 2024.
[6] Apache Software Foundation, "Apache Kafka Documentation," [Online]. Available: https://kafka.apache.org/documentation/
[7] Amazon Web Services, "AWS Lambda Developer Guide," [Online]. Available: https://docs.aws.amazon.com/lambda/latest/dg/
[8] Amazon Web Services, "Amazon Athena User Guide," [Online]. Available: https://docs.aws.amazon.com/athena/latest/ug/
[9] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in Proc. IEEE ICDM, 2008, pp. 413–422.
[10] I. D. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in Proc. ICISSP, 2018, pp. 108–116.
[11] S. M. Lundberg and S. I. Lee, "A unified approach to interpreting model predictions," in Proc. NeurIPS, 2017, pp. 4765–4774.
[12] Amazon Web Services, "AWS Glue Developer Guide," [Online]. Available: https://docs.aws.amazon.com/glue/latest/dg/



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY