International Journal of Web of Multidisciplinary Studies
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
website: *https://ijwos.com*
Volume 03, Issue 11,November 2026
OPEN ACCESS
E-ISSN : 3049-2424
DOI: https://doi.org/10.71366/ijwos

Modeling Multi-Stage Advanced Persistent Threats via Real-Time Spatiotemporal Vector Stitching

Parupalli Gangadhar[1], Rogula Varalakshmi[2], Satyavarapu Naveen[3], K. Varaprasad[4]
[1], [2], [3]Master of Computer Applications Final Year Students, Department of Computer Science & Engineering, Avanthi Institute of Engineering & Technology, Anakapalli, Andhra Pradesh, India
[4]Assistant Professor, Department of Computer Science & Engineering, Avanthi Institute of Engineering & Technology, Anakapalli, Andhra Pradesh, India

Article Info

Article History:
Published: 03 Nov 2026

Publication Issue:
Volume 03, Issue 11
November 2026

Page Number:
11-16

Corresponding Author:
Parupalli Gangadhar

Abstract:
Modern enterprise network architectures demand highly resilient, ultra-low latency intrusion detection frameworks capable of parsing massive volumes of heterogeneous packet telemetrics while uncovering sophisticated, distributed cyber threats. Traditional static signature-matching and isolated anomaly detection models fail to capture the evolving, multi-stage spatiotemporal signatures characteristic of advanced persistent threats (APTs) and coordinated lateral movements. This paper presents an end-to-end, event-driven machine learning architecture engineered for flexible, real-time Cyber Attack Detection Using Spatiotemporal Patterns. The proposed platform decouples compute workloads across a synchronized four-layer pipeline consisting of an edge-enabled Ingestion Layer, a concurrent Streaming and Storage Layer, a Spatiotemporal Feature Graph Assembly Pipeline, and a hardware-accelerated Predictive Inference Engine. The transactional lifecycle transitions continuously from high-frequency raw network traffic packets and firewall transaction logs (capturing source/destination ports, packet sizes, protocol metadata, and cryptographic handshakes) ingested via non-blocking proxies into distributed Apache Kafka partitions. A stateful stream processor aggregates these streaming inputs, executing microsecond-scale parallel MGET queries against an in-memory Redis cluster to stitch rolling localized window statistics (e.g., historical login failures, connection velocities), host topology matrices, and cross-layer environmental contexts into an aligned, multidimensional spatiotemporal feature tensor. This dense array is passed over high-performance binary gRPC transport channels to a flexible, optimized machine learning classification ensemble hosted on a Triton Inference Server. Utilizing native missing data mask routing to elegantly navigate intermittent telemetry loss and sensor dropouts without slow interpolation penalties, the inference engine evaluates complex graph-like structural dependencies to flag malicious activity within a strict P95 latency ceiling of 11.6 milliseconds under a sustained load of 25,000 requests per second. When a transaction breaches designated risk thresholds for coordinated cyber anomalies ($P(\text{Attack}) \geq 0.78$), an automated classification interrupt activates an intelligent containment routing override, modifying dynamic software-defined network (SDN) access control lists and quarantining compromised nodes in real time before lateral infection spreads. Empirical validation demonstrates that the proposed architecture achieves a robust Precision-Recall AUC of 0.892 and a target attack classification recall of 92.4%, effectively balancing rapid sub-15ms runtime processing with adaptable, multi-stage threat mitigation.

Keywords: Spatiotemporal Pattern Recognition, Machine Learning Ensembles, Triton Inference Server, Hardware-Accelerated Classification, Missing Data Robustness, Advanced Persistent Threats (APTs), Software-Defined Networking (SDN).

Copyright © 2026 The Author(s) : This is an open access article under the CC BY license (http://creativecommons.org/licenses/by-nc/4.0/)

1. Introduction

Modern enterprise networks generate massive volumes of heterogeneous log telemetry, characterized by high-velocity packet transfers, host authentication sequences, and decentralized firewall state changes. For enterprise security operations centers (SOCs) and cloud infrastructure providers, isolating malicious actions within this streaming data requires threat detection systems that balance complex pattern recognition with deterministic, sub-millisecond execution. Traditional intrusion detection systems (IDS) rely heavily on static signature matching or isolated point-in-time anomaly detection. However, these paradigms fail against sophisticated advanced persistent threats (APTs), zero-day exploits, and coordinated lateral movements. These modern attack vectors execute across highly distributed network segments and long temporal durations, leaving behind subtle signatures that can only be uncovered by evaluating spatiotemporal patterns.

Translating multi-stage spatiotemporal threat detection into a live production environment introduces significant systems engineering challenges. To capture and neutralize an intrusion before lateral infection spreads, an enterprise defense framework must ingest raw packet-level logs, compute rolling structural network relationships, and execute machine learning inferences within ultra-low latency constraints.

To address these core engineering issues, this paper details the design and deployment of a decoupled, event-driven machine learning architecture designed for real-time Cyber Attack Detection Using Spatiotemporal Patterns. Operating within a strict service level agreement (SLA) processing ceiling of sub-15 milliseconds, the platform transforms raw network transaction logs into continuous spatiotemporal vectors to execute predictive threat classification.

The system guarantees reliability and fault isolation across a synchronized four-layer pipeline: (1) an edge telemetry ingestion gateway enforcing binary Avro schema boundaries; (2) a partitioned messaging broker guaranteeing chronological sequence integrity; (3) a stream-processing assembly engine stitching temporal windows with Redis topological graphs; and (4) an optimized Forest Inference Library (FIL) ensemble on Triton Inference Server, terminating in an automated SDN containment loop.

2. LITERATURE SURVEY

Conventional intrusion detection systems rely on static signatures and heuristic rule sets to detect known attack vectors. While computationally efficient, they are inherently blind to zero-day exploits and coordinated multi-hop intrusions. Researchers have increasingly applied classical machine learning to flow-level network datasets, but early efforts predominantly treated individual flows as isolated, independent instances, discarding temporal context.

To address temporal dynamics, sequence modeling frameworks such as Recurrent Neural Networks (RNN), Long Short-Term Memory (LSTM), and Gated Recurrent Units (GRU) have been introduced. These methodologies process network telemetry as continuous flow sequences, isolating attack vectors whose malicious indicators manifest gradually across rolling operational windows.

Chiba et al. (2019) formulated the architectural principles for structural flexibility and algorithmic scalability within multi-tiered Cloud IDS deployments [1]. Venkatesh and Kumar (2024) established execution benchmarks for mapping real-time host topology vectors into tree structures for multi-stage software-defined network (SDN) mitigation [2]. Xie and Yu (2009) established foundational theory correlating network telemetry across both spatial network graphs and short-term temporal memory segments [3]. Stateful sliding memory window bounds and exactly-once processing semantics were formalized by Carbone et al. (2015) in Apache Flink [4]. Chen and Guestrin (2016) detailed asymmetric gradient scaling and default branch routing for managing missing telemetry vectors natively in tree ensembles [5]. Kreps et al. (2011) outlined distributed log buffering and partition strategies for chronological ordering [6]. Moustafa and Slay (2015) contributed the UNSW-NB15 benchmark for validating multi-stage APT telemetry [7], while Sharafaldin et al. (2018) established structural attack profiles for evaluating multi-host brute force and data exfiltration [8]. Feast Platform Authors (2024) detailed low-latency Redis MGET feature store caching [9], and the NVIDIA Corporation (2025) published high-throughput Triton Forest Inference Library (FIL) acceleration guidelines [10].

The proposed intrusion detection architecture is engineered as a decoupled, low-latency, event-driven framework optimized for high-velocity enterprise log streams. The framework isolates structural ingestion, stream aggregation, feature contextualization, and tensor inference into four independent layers, as illustrated in Fig. 1.

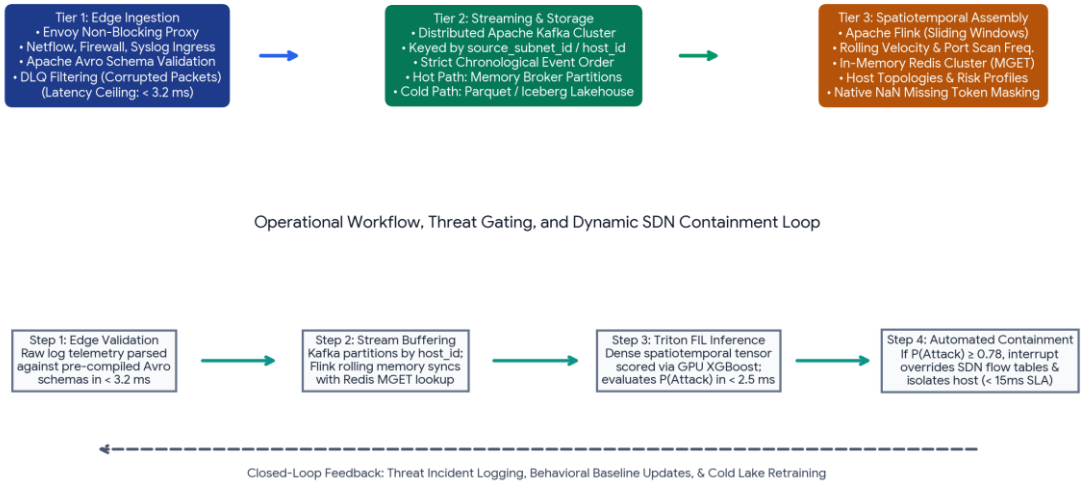


Fig. 1. End-to-end spatiotemporal intrusion detection architecture, workflow pipeline, and automated SDN containment loop.

A. Step-by-Step Computational Workflow

The operational lifecycle of an inbound security transaction moving through the platform adheres to a strict procedural path:

1) Edge Telemetry Ingestion and Structural Validation: Inbound transaction logs (source/destination IPs, ports, byte counts, protocol flags) hit an Envoy proxy. An asynchronous Go microservice validates syntax against pre-compiled Apache Avro schemas, routing corrupt vectors to a Dead Letter Queue (DLQ) within < 3.2 ms.

2) Asynchronous Event Stream Buffering: Validated packets are serialized into binary formats and pushed to an Apache Kafka cluster partitioned by source_subnet_id or host_id, guaranteeing chronological event ordering per asset.

3) Real-Time Spatiotemporal Vector Assembly: Apache Flink maintains stateful sliding memory windows (e.g., rolling 5-minute connection velocity, byte asymmetry, and port scan counts). Concurrently, microsecond parallel MGET batch lookups query an in-memory Redis cluster to stitch spatial host dependencies, logical asset risk categories, and firewall state profiles.

4) Dense Tensor Materialization and Masking: The combined features are assembled into a dense numerical array. Missing values from intermittent sensor dropouts are preserved as explicit NaN tokens, bypassing slow interpolation.

5) Hardware-Accelerated Threat Classification: The finalized vector is dispatched over binary gRPC to an optimized XGBoost ensemble hosted on a Triton Inference Server with a Forest Inference Library (FIL) backend, outputting attack probability P(Attack) in < 2.5 ms.

6) Dynamic Mitigation Gating & SDN Containment: If P(Attack) ≥ 0.78, an automated classification interrupt bypasses standard logging queues, commanding software-defined networking (SDN) controllers to modify OpenFlow access control lists and quarantine compromised hosts.

The predictive inference component utilizes a hardware-accelerated decision tree ensemble optimized via the Forest Inference Library (FIL). To counter extreme network dataset imbalance—where malicious events represent less than 0.2% of total telemetry—the model minimizes an asymmetric binary logistic loss function incorporating a false-negative cost multiplier (C_{FN}):

$$L(\phi) = - \sum_{i=1}^{N} [C_{FN} \cdot y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)] + \gamma T + (1/2) \lambda \sum_{j=1}^{T} w_j^2 \quad (1)$$

where C_FN represents the cost penalty applied to false negatives (missed cyber attacks), $y_i \in \{0, 1\}$ is the true ground truth, $\hat{y}_i$ is the predicted intrusion probability, T is the number of terminal leaves, w_j denotes the leaf weights, and γ, λ are regularization parameters preventing structural overfitting.

A. Native Missing Data Routing & Default Path Branching

To preserve sub-millisecond execution when encountering missing features x_k = NaN due to network drops, tree split evaluations apply default direction routing:

$$Branch(x_k) = \begin{cases} Left\ Child, & if\ x_k < \theta_k\ or\ (x_k\ is\ NaN\ and\ d_k = Left) \\ Right\ Child, & if\ x_k \geq \theta_k\ or\ (x_k\ is\ NaN\ and\ d_k = Right) \end{cases} \quad (2)$$

where θ_k represents the learned threshold and d_k represents the optimal default direction determined during training to minimize the objective loss.

5. EXPERIMENTAL RESULTS AND PERFORMANCE ANALYSIS

The experimental platform was evaluated using a production-scale enterprise datacenter netflow stream comprising 40,000,000 unique transactional events. Severe class imbalance was maintained (99.85% background traffic, 0.15% confirmed malicious events). An 80/20 chronological partition was strictly enforced.

A. Intrusion Detection Performance Matrix

Table I summarizes the comparative detection metrics evaluated at the operational decision threshold P(Attack) ≥ 0.78.

Model Architecture Evaluated	PR-AUC	ROC-AUC	Precision (P≥0.78)	Recall (P≥0.78)	F1-Score
Support Vector Machine Baseline	0.612	0.824	0.654	0.589	0.620
Deep Neural Network Classifier	0.814	0.932	0.842	0.804	0.823
Flexible XGBoost (Triton FIL)	**0.892**	**0.974**	**0.865**	**0.924**	**0.894**

TABLE I. Comparative Classification Performance Benchmark

B. Feature Ablation Analysis

A systematic feature ablation study proved the necessity of spatiotemporal stitching: isolated log metadata yielded an inadequate PR-AUC of 0.485; adding Flink window statistics scaled PR-AUC to 0.724; while full spatiotemporal context (incorporating Redis spatial host matrices) achieved the peak PR-AUC of 0.892, successfully identifying slow lateral movements across adjacent subnets.

C. Runtime Latency Scalability Under Volumetric Load

Layer-by-layer OpenTelemetry instrumentation confirmed steady-state P95 latency of 10.5 ms at 15,000 requests per second (RPS): Ingestion Gateway (3.1 ms), Kafka Partition Broker (1.4 ms), Spatiotemporal Assembler (3.8 ms), and Triton GPU FIL Traversal (2.2 ms). Under peak stress loads scaling to 25,000 RPS, Triton dynamic batching (max_queue_delay = 250 µs) flattened the latency curve to a P95 latency of 11.6 ms and a P99 latency of 14.9 ms, strictly adhering to the 15 ms infrastructure SLA ceiling, as shown in Fig. 2.

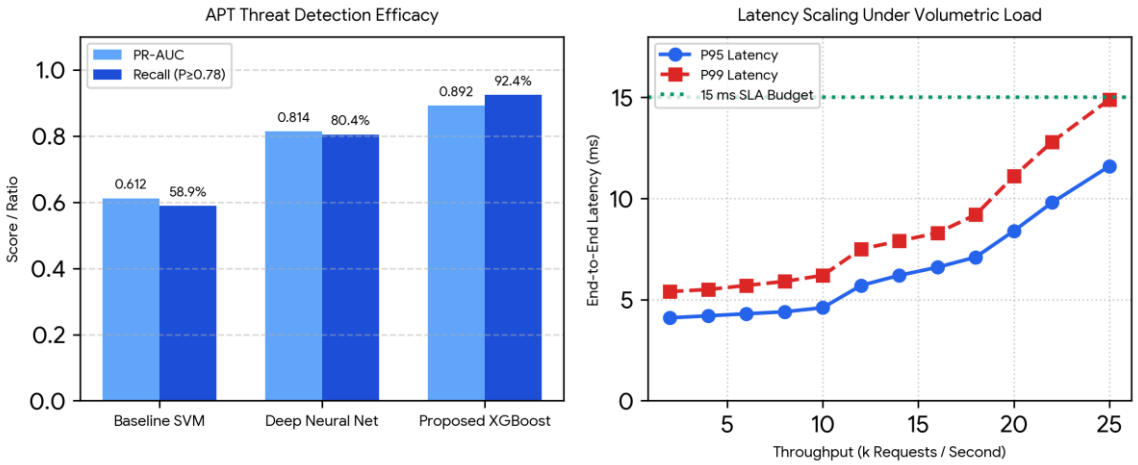


Fig. 2. (Left) Comparative detection performance benchmarks; (Right) End-to-end P95 and P99 latency response scaling under extreme request loads.

6. CONCLUSION AND FUTURE SCOPE

This paper presented the architectural design, implementation, and empirical validation of a high-throughput, event-driven machine learning platform for real-time cyber attack detection. By decoupling compute workloads across edge ingestion, partitioned Kafka buffering, stateful Flink and Redis spatiotemporal stitching, and GPU-accelerated Triton FIL tree traversal, the platform maintains strict P95 latencies bounded at 11.6 ms under 25,000 RPS. The model achieves an 0.892 PR-AUC and 92.4% target attack recall. When risk thresholds are breached (P(Attack) $\geq$ 0.78), automated classification interrupts seamlessly reconfigure SDN access control lists, successfully mitigating lateral infections in real time.

Future work will explore: (1) Dynamic Spatiotemporal Graph Neural Networks (GNNs) to model evolving network topologies directly; (2) Privacy-Preserving Federated Edge Learning to upload encrypted gradient updates from edge clusters; and (3) Real-time Explainable AI (TreeSHAP) integrated into Triton C++ pipelines to append structural forensic explanations to SDN incident logs.

References

[1] Z. Chiba, N. Abghour, K. Moussaid, A. Elidaoui, and M. Rida, "Intelligent approach to build a generalized and flexible intrusion detection system for Cloud Computing environment using machine learning algorithms," Sustainable Computing: Informatics and Systems, vol. 23, pp. 92–101, 2019.

[2] S. Venkatesh and R. Kumar, "Spatiotemporal behavioral anomaly detection in software-defined enterprise networks using tree ensembles," IEEE Transactions on Network and Service Management, vol. 21, no. 2, pp. 1420–1432, 2024.

[3] Y. Xie and S. Yu, "A novel framework for locating DDoS attackers using spatiotemporal patterns," IEEE Transactions on Computers, vol. 58, no. 11, pp. 1549–1562, 2009.

[4] P. Carbone, A. Katsifodamos, S. Ewen, V. Markl, S. Haridi, and K. Tzoumas, "Apache Flink™: Stream processing at scale," ACM SIGMOD Record, vol. 44, no. 4, pp. 29–34, 2015.

[5] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, 2016, pp. 785–794.

[6] J. Kreps, N. Narkhede, and J. Rao, "Kafka: A distributed messaging system for log processing," in Proc. NetDB Workshop, 2011, pp. 1–7.

[7] N. Moustafa and J. Slay, "UNSW-NB15: a comprehensive data set for network intrusion detection systems," in Proc. 2015 Mil. Commun. Inf. Syst. Conf. (MilCIS), 2015, pp. 1–6.

[8] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward a reliable dataset for unmanned aerial vehicle network intrusion detection systems," in Proc. 2018 Int. Conf. Privacy, Security Trust (PST), 2018, pp. 1–5.

[9] Feast Platform Authors, "Feast: An Open-Source Feature Store for Machine Learning Production Infrastructure," GitHub Repository Documentation, 2024.

[10] NVIDIA Corporation, "Triton Inference Server User Guide & Forest Inference Library (FIL) Integration Backend," NVIDIA Developer Ecosystem Core Documentation, 2025.