TrustGraph: A Heterogeneous GNN for Dynamic Zero-Trust Policy Enforcement in Microservices

Nurmyrat Amanmadov[1], Jemshit Iskanderov[2], Tarlan Abdullayev[3]
University of Washington, Seattle, USA[1]
Touro University, New York, USA[2]
Northeastern University, Boston, USA[3]

***Abstract*—Securing cloud microservices requires a unified understanding of how services behave, authenticate, and interact in real time. Unlike existing methods that analyze telemetry signals in isolation, this work presents a heterogeneous graph-based Zero-Trust framework that represents microservices using multi-modal telemetry—logs, metrics, traces, and authentication flows—embedded directly into graph nodes and edges. A Graph Neural Network architecture with attention captures risk propagation across service dependencies, while a joint anomaly detection and trust computation mechanism generates dynamic trust scores with temporal decay to support continuous verification. These trust signals drive real-time dynamic policy enforcement capable of denying or restricting suspicious interactions with minimal operational overhead. Experiments on the TrainTicket, Sock Shop, and DeathStarBench benchmarks show strong performance, achieving 97.2% accuracy, 98.1% recall, and 0.987 AUC on TrainTicket, with consistent results across the other datasets and latency overhead below 3.2 ms. Robustness tests demonstrate accuracy above 95.8% under noisy logs, delayed traces, and authentication failures. Ablation and SHAP analyses confirm that leveraging multiple telemetry modalities—especially authentication data—is critical for accurate detection and trust scoring. These findings show that multi-modal heterogeneous graph modeling, coupled with integrated anomaly-to-policy decision pipelines, provides an effective foundation for Zero-Trust security in cloud-native microservices.**

Keywords*—Graph neural networks; zero-trust security; microservices; anomaly detection; heterogeneous graphs; multi-modal telemetry; dynamic policy enforcement

I. Introduction

Cloud computing environments have increasingly shifted toward microservice architectures, where applications are decomposed into loosely coupled services [1]. This design improves modularity and scalability but introduces new challenges in reliability, security, and anomaly detection [2]. Microservices communicate frequently across distributed infrastructures, which expands the attack surface and creates opportunities for unauthorized access or service misuse [3]. Conventional perimeter-based security models are not suited for such settings because they assume trusted internal zones, while microservices interact dynamically across diverse platforms [4]. The distributed and transient nature of services calls for approaches that can capture complex interactions while maintaining protection against internal and external threats [5].

Zero-Trust Architecture (ZTA) has emerged as a suitable paradigm to address these challenges [6]. ZTA removes implicit trust and applies continuous verification of services, users, and communication flows [7]. This approach combines anomaly detection, runtime trust scoring, and adaptive policy enforcement to secure interactions at fine granularity [8]. In microservice systems, ZTA introduces mechanisms to restrict or deny suspicious requests even when they originate from within the network [9]. Such continuous monitoring aligns with the dynamic behavior of cloud deployments, where trust must be recalculated as services update, scale, or interact with new components [10]. ZTA provides a structured framework for controlling risks in environments where static boundaries are no longer sufficient [11].

Graph-based models have gained attention as a means to represent dependencies in microservice environments [12] [13]. Unlike traditional methods that treat services in isolation, graph-based approaches capture relationships between nodes and edges representing services, requests, and identities [14]. This relational modeling is crucial for analyzing distributed execution patterns, detecting abnormal behavior, and assigning trust in real time [15]. The integration of multiple telemetry sources, including logs, metrics, traces, and authentication records, strengthens the ability to represent both operational and security contexts [16]. Together, graph-based methods and Zero-Trust principles offer a pathway to address the growing need for effective anomaly detection and adaptive policy enforcement in modern cloud systems [17].

Most existing microservice environments face a research problem rooted in the gap between security requirements and available detection methods [18]. While anomaly detection techniques and monitoring systems exist, they often lack the ability to represent complex dependencies across distributed services [19]. Isolated tools struggle to capture the combined effect of operational behaviors, access flows, and trust decisions [20]. This creates a vulnerability in cloud deployments, where malicious activity may pass unnoticed or trusted connections may be abused [21]. The research challenge lies in unifying detection and policy enforcement under a framework that reflects the realities of microservice interactions [22].

To address this, researchers have introduced a variety of methods [23]. Some works focus on mining logs or metrics using statistical or embedding-based approaches, which can identify outliers in specific datasets [24]. Others apply anomaly detection at the level of traces or flows, which highlights unusual patterns of interaction [25]. These approaches contribute useful insights, yet they are limited when applied in isolation [26]. Logs may reveal system errors but overlook authentication misuse. Metrics may show performance degradation but fail to detect authorization bypass [27]. Traces highlight interaction delays but cannot capture runtime trust

shifts. Relying on a single modality leaves gaps that attackers can exploit.

In parallel, several studies have suggested frameworks that combine anomaly detection with Zero-Trust enforcement [28]. These describe runtime trust scoring, authorization checks, and adaptive policies [29]. While these approaches advance the discussion, they often depend on simplified models of service interaction [30]. Graph structures are sometimes restricted to service-to-service calls without incorporating authentication, authorization, or contextual data [31]. Many solutions remain tied to specific benchmarks, which limits their scalability to real-world workloads [32]. In addition, explainability and privacy-preserving training are frequently overlooked, leaving questions about transparency and applicability in multi-tenant environments [33].

Despite these advances, existing approaches largely treat security and operations as separate concerns. Most graph-based methods focus on topology or call chains without integrating authentication flows as first-class entities in the graph structure. Conversely, Zero-Trust frameworks often rely on static policies or isolated anomaly detectors that lack the context of service dependencies. A key gap remains in establishing an end-to-end pipeline where multi-modal telemetry—especially authentication data—directly informs dynamic trust scores that automatically update enforcement policies in real time.

The core contribution of this work is a unified framework that bridges this gap. Specifically, this paper contributes:

- Heterogeneous Graph Construction: We propose a method to represent microservices where authentication flows and operational metrics are embedded as distinct node and edge features, enabling the GNN to learn the interplay between performance and security.
- Joint Anomaly-Trust Mechanism: We introduce a dynamic trust scoring algorithm that combines reconstruction error with temporal decay, allowing trust to degrade naturally over time in the absence of positive reinforcement—a key tenet of Zero-Trust.
- Closed-Loop Policy Enforcement: We demonstrate an end-to-end pipeline where trust scores are mathematically linked to policy optimization, automating the transition from anomaly detection to access control (Allow/Restrict/Deny) with minimal latency.

Building on these contributions, the following research questions are formulated to guide the study.

1) RQ1: How can heterogeneous graphs be constructed to represent microservice environments by integrating both operational and security features?
2) RQ2: In what ways can anomaly detection and trust computation be performed using multimodal telemetry to distinguish between benign and anomalous behaviors?
3) RQ3: How can detection outcomes and trust scores be systematically linked to policy enforcement for allowing, restricting, or denying service interactions in real time?

The remainder of this paper is organized as follows. Section II reviews prior works on anomaly detection, Zero-Trust enforcement, and graph-based methods for microservices. Section III presents the graph-based framework, including system modeling, anomaly detection, trust computation, and policy enforcement. Section IV outlines the benchmarks, datasets, and experimental configuration. Section V reports the findings, including performance metrics and robustness analysis. Section VI provides a deeper discussion on the implications of risk propagation and trust decay. Finally, Section VII summarizes the contributions and highlights directions for future work.

II. Literature Review

The adoption of microservice architectures in cloud environments created new challenges for security, reliability, and anomaly detection. Traditional perimeter-based models were insufficient in handling distributed service interactions, which made ZTA an important direction for research. Scholars developed methods that combined anomaly detection, policy enforcement, and runtime trust scoring to address these challenges. Graph Neural Networks (GNNs) were introduced as suitable models since they captured relationships among services, requests, and identities. In addition, studies applied multimodal monitoring data, federated learning, and explainable frameworks to improve detection accuracy and interpretability. Several works used benchmarks such as Train-Ticket and Sock-Shop to confirm proposed methods with quantitative metrics. Other contributions emphasized architectural frameworks in service mesh and multi-cloud scenarios, describing trust evaluation and policy transparency. Collectively, the literature highlighted advances in anomaly detection, Zero-Trust enforcement, and GNN-based security for microservices.

Raeiszadeh et al. [34] developed an asynchronous federated learning method for anomaly detection in microservice systems using traces and logs from the TrainTicket benchmark. The method combined GNNs with Positive and Unlabeled learning and introduced local training with a span causal graph. The model achieved improvements of about 4% in F1 compared with state-of-the-art anomaly detectors and reduced detection latency with controlled overhead.

Cinque et al. [35] suggested Micro2vec to mine numeric log embeddings for anomaly detection in Clearwater IMS and Air Traffic Control case studies. Their classifiers, including autoencoders and feed-forward networks, achieved accuracy of 97% with F1 values around 0.961. The approach highlighted the role of embeddings for distinguishing anomalies in log data.

Parkhomenko et al. [36] used Zero-Trust principles and machine learning for detecting authorization anomalies in microservices. Their framework described continuous verification of requests and anomaly detection in authentication flows. The study was more conceptual and reported qualitative improvements with reduced false positives. Alboqmi et al. [37] defined a runtime trust evaluation system in a service mesh environment using Train-Ticket deployed on Istio. The framework combined vulnerability scanning results with service mesh telemetry to produce trust scores at runtime. The outcomes were presented as architectural contributions with qualitative benefits for runtime risk assessment. Both studies described Zero-Trust concepts for anomaly and trust monitoring, one addressing authorization attacks and the other runtime trust scoring.

Zhao, Xiao and Tao [38] applied a multi-scale dynamic GNN for anomaly detection using three industrial microservice datasets. Their model adapted neighborhood sizes and temporal depth, showing improved accuracy compared with conventional graph models. Ablation results described the importance of multi-scale design choices. Fang et al. [39] described a method extending GraphSAGE with attention to locate anomalies in Kubernetes-based microservices. Their results showed P@1 of 76%, P@3 of 88%, and P@5 of 89%, surpassing baseline GraphSAGE values of 54%, 72%, and 79%. The approach localized faulty services more effectively, providing interpretable results for operators. Both studies presented graph-based models for anomaly detection and localization with measurable performance gains. Table I summarized studies on microservice security. Raeiszadeh et al. 1 applied federated GNNs, while Cinque et al. 3 used log embeddings. Alboqmi & Gamble 6 described runtime trust in service mesh, and Zhao, Xiao & Tao 7 developed a multi-scale dynamic GNN. Fang et al. 8 and Nobre et al. 9 further extended detection with measurable accuracy.

Nobre et al. [40] used Multi-Layer Perceptrons for detecting anomalies in Sock-Shop and TrainTicket benchmarks with fault injection. Their service-level dataset showed accuracy of 97%, precision of 95%, recall of 98%, and F1 of 97%, while maintaining false positive rates near 0.01. Zhao et al. [43] developed IM-GNN to recommend microservice orchestration workflows using interface matching and BERT embeddings with GNNs. The results on Amazon SageMaker showed Hit@10 scores of 41.45 against 36.23 and NDCG@10 of 31.23 against 28.44. Although IM-GNN targeted orchestration rather than security, it demonstrated the effectiveness of graph construction in improving results. Both studies described neural approaches with high performance, one focusing on anomaly detection and the other on orchestration accuracy.

Anderson [44] described the role of Explainable Artificial Intelligence in Zero-Trust API gateways on Microsoft Azure. The framework integrated LIME, SHAP, and causal reasoning for interpreting access control decisions. Results described improved transparency and auditability with some latency overhead. Panchumarthi [45] defined the NCAF Zero-Trust microservice fabric combining microsegmentation with runtime authorization strategies. The study emphasized architectural design and described conceptual benefits rather than empirical evaluation. Both works contributed to the role of Zero-Trust in cloud authorization, focusing on explainability and structural frameworks. The findings highlighted transparency, centralized authorization, and policy consistency without numerical benchmarks.

Cherukupalle [46] introduced a Zero-Trust hybrid framework for cloud-IoT testbeds, integrating blockchain and AI monitoring. The approach reduced detection latency by 30% and blocked more intrusion attempts. Madhura G K [47] applied AI trust scoring for Zero-Trust in microservices with synthetic datasets and reported trust prediction accuracy near 92%. Konakanchi [48] developed predictive cyber resilience for healthcare microservices, reporting breach reduction of 90% and mean time to detect shortened by 40%. Collectively, these contributions described AI-driven and graph-based methods improving anomaly detection and resilience.

Zhuwankinyu et al. [49] suggested a graph-based Zero-Trust framework for distributed storage, achieving improved detection accuracy with minimal latency. Kodakandla [50] described Zero-Trust enforcement in Kubernetes microservices, emphasizing compliance improvements and policy consistency as qualitative outcomes. Adanigbo et al. [51] applied a Zero-Trust framework for multi-cloud environments, reporting higher resilience and trust metric scores. Reddy [52] introduced microsegmentation with AI-driven policy engines in hybrid cloud testbeds, blocking 91% of lateral movements while adding about 1.5 ms in latency. Shonubi and Adelere [53] developed AI-augmented resilience frameworks across SDN and microservices, reducing outage times by 87% and detection times from 18 minutes to 45 seconds. Kokku [54] described a synthesis of 29 studies integrating Zero-Trust, federated learning, and AI trust models, highlighting improvements across detection accuracy and trust management.

Although much work had been done on anomaly detection and Zero-Trust enforcement in microservices, clear gaps remained. Most methods relied on logs, metrics, or traces in isolation, with limited use of multimodal signals in real-time settings. GNN approaches often used simplified service graphs, while heterogeneous graphs that included authentication, authorization, and data flows were rarely applied. Zero-Trust frameworks were introduced, yet few connected anomaly scores to policy synthesis for continuous microsegmentation. Many studies achieved high accuracy on synthetic benchmarks such as Train-Ticket and Sock-Shop, but large-scale workloads were less examined. Federated learning appeared in some works, but privacy-preserving training across multi-tenant clusters was not fully described. Explainability was discussed in access control, but graph-level explanations in microservice security were largely absent. Latency and scalability trade-offs in Zero-Trust enforcement also lacked thorough analysis, leaving room for integrated graph-based, explainable, and federated approaches.

III. Proposed Methodology

The methodology addressed the problem of anomaly detection and Zero-Trust enforcement in microservice environments by formulating the system as a graph-based learning problem. Microservices were represented as nodes, and their interactions were expressed as edges enriched with operational and security features. GNNs were used to propagate information across the service dependency graph, allowing both local and global risks to be captured. Anomaly detection was carried out through reconstruction-based scoring, while trust values were derived from anomaly signals and authentication results. Policies were then optimized to restrict, allow, or deny service interactions based on computed trust. The following subsections describe system modeling, embedding, anomaly detection, trust modeling, and policy enforcement, each supported by mathematical formulations and algorithms to demonstrate the integration of learning with Zero-Trust security principles.

The architecture in Fig. 1 illustrated the integration of microservice telemetry with graph construction. Logs, metrics, traces, and authentication records were normalized into a heterogeneous graph. Graph Neural Networks propagated risk information through message passing and attention, producing embeddings for nodes and edges. Anomaly scores and trust values were computed from these embeddings to capture

TABLE I. SUMMARY OF 22 REVIEWED WORKS ON ANOMALY DETECTION, GNNS, AND ZERO-TRUST IN MICROSERVICES, INCLUDING DATASETS, METHODS, LIMITATIONS, AND RESULTS

Ref	Dataset used	Methodology (brief)	Limitations	Evaluation results
[12]	Custom dataset & TrainTicket	HG-PAD, Conditional VAE	Not multi-modal, not dynamic	Avg@5 of 0.70 to 0.88 for RCL; F1=92%, Rec=98%, Prec=87% for AD
[34]	TrainTicket traces+logs	ART-FL: Async FL + GNN	Perf focus; comms privacy not covered	+4% F1 over SOTA; latency/overhead reported
[35]	Clearwater IMS & ATC logs	Numeric embeddings + ML classifiers	Synthetic anomalies; log-only	Acc 97%; Autoencoder P=0.969, R=0.953, F1=0.961
[36]	Simulated auth flows	Zero-Trust verification + anomaly detection	Workshop-style; limited quant	Prec=95.2%; F1=95.9%;
[37]	Train-Ticket on Istio	Vulnerability-driven trust evaluation	Architectural; limited metrics	Qualitative trust/risk improvements
[38]	Industrial datasets (3)	MSDG: Multi-scale dynamic GNN	Dataset names truncated	Outperforms baselines; ablations confirm design
[39]	Kubernetes (Sock Shop), Mocker	GraphSAGE+Attention RCA	Focus RCA not prevention	P@1=76%; P@3=88%; P@5=89%
[40]	Sock-Shop & TrainTicket	MLP anomaly detection	Needs labels; tied to benchmark	Acc=97%; Prec=95%; Rec=98%; F1=97%; FPR=0.01
[41]	Custom dataset	LSTM, GAT	Doesn't reflect production-level microservices	Acc=95%, Rec=89% for 6:4 data split; Acc=98%, Rec=92% for 9:1 data split;
[42]	TrainTicket	GNN-SVDD	Log based	Prec=93%, Rec=97%, F1=95%
[43]	SageMaker & Comprehend	IM-GNN: interface match + GNN	Orchestration rather than security	Hit@10=41.45; NDCG@10=31.23
[44]	Azure API Mgmt+AD	XAI (LIME, SHAP) for Zero-Trust APIs	Conceptual; perf overhead	Transparency improved; latency overhead
[45]	Conceptual fabric	Federated, ZTA	No dataset	Qualitative only
[46]	Cloud-IoT testbed	Zero-Trust+Blockchain+AI	Prototype scale	Detection latency ↓30%; breaches reduced
[47]	Synthetic IoT workloads	Zero-Trust+AI trust scoring	Conceptual heavy	Trust score accuracy 92%
[48]	HealthTech microservices	FL, SAN, adversarial ML resilience	Health domain only	Breach reduction 90%; MTTD ↓40%
[49]	Distributed storage graphs	Graph-based Zero-Trust storage	Focus storage not services	Improved accuracy; minimal latency
[50]	Kubernetes workloads	Zero-Trust enforcement	Design-level	Better compliance (qualitative)
[51]	Multi-cloud setups	Zero-Trust framework	Review-style	Trust metrics; resilience scores
[52]	Hybrid cloud testbed	Microsegmentation+AI Zero-Trust	Policy complexity	91% lateral moves blocked; latency +1.5ms
[53]	Financial+HealthTech	AI-aug, SDN+anomaly detection	IoT focused; overhead	DDoS MTTM 18m→45s; outages ↓87%
[54]	Literature (29 papers)	Hybrid AI+Zero-Trust synthesis	Review; no dataset	Summarizes ↑accuracy in arc-fault, IoT trust

both behavioral and security contexts. Finally, policies were synthesized and enforced at gateways, meshes, and Kubernetes layers to implement Zero-Trust decisions in real time.

A. System Modeling

1) Graph Representation of Microservices:

$$G = (V, E, X_V, X_E) \tag{1}$$

Eq. (1) defined the microservice architecture as a heterogeneous graph. Nodes represented services, edges represented calls, node features encoded operational telemetry, and edge features described communication and authentication. This formulation allowed the model to capture both performance and security dynamics under Zero-Trust principles.

2) Weighted Adjacency Matrix:

$$A_{uv} = \begin{cases} w_{uv}, & \text{if } (u,v) \in E \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

Eq. (2) introduced weighted adjacency to reflect risk factors such as latency or access violations. Unlike binary graphs, this allowed communication paths to exist while still carrying risk. This encoded the Zero-Trust assumption that no connection should be inherently trusted.

The construction process, which extracts node features from metrics and logs for all nodes $v \in V$ and computes edge features from authentication records and latency for all edges $(u, v) \in E$, is summarized in Algorithm 1.

Algorithm 1 Graph Construction for Microservice Zero-Trust

Input : Microservice system telemetry (Logs, Metrics, Traces), Authentication Records

Output: Telemetry-enriched Graph $G = (V, E, X_V, X_E)$

Initialize the set of Vertices V (microservices/pods) and Edges E (communication links) Initialize Node Feature Matrix X_V and Edge Feature Matrix X_E

foreach *node* $v \in V$ **do**
  Extract operational telemetry x_v from Metrics and Logs Store x_v in X_V

foreach *edge* $(u, v) \in E$ **do**
  Compute edge features x_{uv} from Traces (latency) and Authentication Records Store x_{uv} in X_E

Construct the Weighted Adjacency Matrix A using the edge features X_E (Eq. III-A2) **return** Graph $G = (V, E, X_V, X_E)$

B. Graph Neural Embedding

1) Node Initialization:

$$h_v^{(0)} = x_v \tag{3}$$

Eq. (3) initialized node embeddings from telemetry. Each service embedding captured CPU usage, request errors, and access results. This raw mapping grounded the learning process in measurable service data.

2) Message Passing:

$$m_{uv}^{(k)} = W^{(k)} \cdot [h_u^{(k)} \parallel x_{uv}] \tag{4}$$

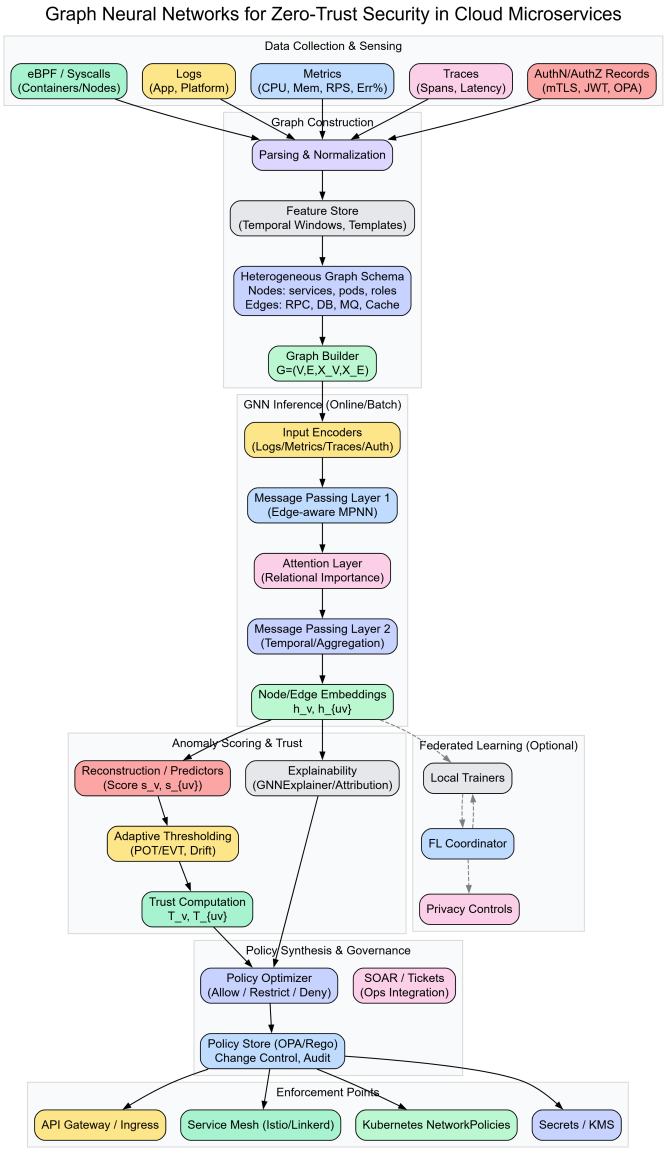

Fig. 1. Architecture of graph neural networks for zero-trust security in cloud microservices.

Eq. (4) defined messages exchanged between services. Embeddings were concatenated with edge features to capture both behavior and security posture. This confirmed risk factors in communication were directly embedded.

3) Attention Weights:

$$\alpha_{uv}^{(k)} = \frac{\exp\left(a^\top [W h_u^{(k)} \parallel W h_v^{(k)} \parallel x_{uv}]\right)}{\sum_{j \in \mathcal{N}(v)} \exp\left(a^\top [W h_j^{(k)} \parallel W h_v^{(k)} \parallel x_{jv}]\right)} \tag{5}$$

Eq. (5) computed attention coefficients. It weighted neighbors differently, with higher weight on risky or abnormal interactions. This aligned GNNs with Zero-Trust by emphasizing suspicious dependencies.

4) Attention Based Node Update:

$$h_v^{(k+1)} = \sigma\left(\sum_{u \in \mathcal{N}(v)} \alpha_{uv}^{(k)} \cdot m_{uv}^{(k)}\right) \tag{6}$$

Eq. (6) showed how updated embeddings selectively prioritized critical edges. Services influenced each other proportionally to trust relevance, reinforcing Zero-Trust monitoring.

The overall iterative process for computing and updating the final embeddings, incorporating neighborhood aggregation and the attention mechanism, is detailed in Algorithm 2

Algorithm 2 GNN Embedding with Attention

Input: Graph G, initial embeddings $h_v^{(0)}$
Output: Final embeddings $h_v^{(K)}$
for $k = 0$ *to* $K-1$ **do**
  for *each node* $v \in V$ **do**
    Compute messages $m_{uv}^{(k)}$ using Eq. 4 Aggregate neighbors with Eq. 6 Update embedding $h_v^{(k+1)}$
Return $h_v^{(K)}$

C. Anomaly Detection

1) Reconstruction Loss:

$$s_v = \|h_v^{(K)} - \hat{h}_v\|_2^2 \tag{7}$$

Eq. (7) defined the anomaly score as reconstruction error. High deviations meant services were behaving abnormally, useful for Zero-Trust validation.

2) Threshold Decision:

$$y_v = \begin{cases} 1, & s_v > \theta \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

Eq. (8) marked services as anomalous if their anomaly score exceeded a threshold. This binary decision supported automated Zero-Trust policy enforcement.

3) Edge Anomaly Score:

$$s_{uv} = \|h_u^{(K)} - h_v^{(K)}\|_2 \tag{9}$$

Eq. (9) computed anomaly scores for edges. This verified service-to-service interactions, detecting unauthorized or risky communication links.

D. Trust Modeling

1) Node Trust Score:

$$T_v = \frac{1}{1 + e^{-(\alpha(1-s_v)+\beta R_v)}} \tag{10}$$

Eq. (10) transformed anomaly scores and resilience into trust values between 0 and 1. This probabilistic trust confirmed continuous verification of services.

2) Edge Trust Score:

$$T_{uv} = \gamma \cdot (1 - s_{uv}) + (1 - \gamma) \cdot P_{auth}(u, v) \tag{11}$$

Eq. (11) balanced edge anomaly with authentication success. It allowed the system to penalize risky communication while rewarding confirmed authentication.

3) Temporal Trust Decay:

$$T_v(t+1) = \lambda \cdot T_v(t) + (1-\lambda)(1-s_v) \tag{12}$$

Eq. (12) defined trust decay over time. A service's trust decreased naturally unless continuously supported by normal activity, reflecting Zero-Trust.

The complete procedure for calculating the dynamic trust values for both services and interactions, including the application of temporal decay, is presented in Algorithm 3

Algorithm 3 Dynamic Trust Computation

Input: Node scores s_v, edge scores s_{uv}, authentication data
Output: Trust values T_v, T_{uv}
for *each node* $v \in V$ **do**
  Compute T_v using Eq. 10 Apply temporal decay Eq. 12
for *each edge* $(u, v) \in E$ **do**
  Compute T_{uv} with Eq. 11
Return trust scores

E. Policy Enforcement

1) Policy Optimization:

$$\min_{\pi} \sum_{v \in V} \big(\ell(y_v, \pi(v)) + \lambda \cdot \max(0, \tau - T_v)\big) \tag{13}$$

Eq. (13) optimized access policies by minimizing classification loss and penalizing low-trust assignments. This mathematical framing integrated GNN learning with Zero-Trust enforcement.

2) Access Control Constraints:

$$\pi(v) \in \{\text{allow}, \text{restrict}, \text{deny}\}, \quad \pi(v) = \text{deny} \ \text{ if } T_v < \tau \tag{14}$$

Eq. (14) constrained policy outcomes to three options. Services below the trust threshold were denied, embodying Zero-Trust continuous verification.

3) Overall Loss Function:

$$\mathcal{L} = \mathcal{L}_{GNN} + \alpha \sum_{v \in V} s_v + \beta \sum_{(u,v) \in E} s_{uv} + \gamma \sum_{v \in V} \max(0, \tau - T_v) \tag{15}$$

Eq. (15) unified GNN training, anomaly detection, and trust penalties. The formulation made learning and Zero-Trust enforcement a single optimization objective.

IV. Experimental Setup

The experimental setup was designed to confirm the proposed framework for anomaly detection and Zero-Trust enforcement in microservices. All experiments were conducted on a Kubernetes cluster with 12 worker nodes, each configured with 32-core Intel Xeon processors and 128 GB of memory. GPU acceleration was available for model training. Monitoring tools including Prometheus, Jaeger, and eBPF probes were integrated to collect continuous telemetry consisting of logs, metrics, traces, and authentication records. Each dataset was deployed in an isolated name space to avoid interference, and uniform resource quotas confirmed consistent experimental conditions across runs.

The first dataset used was the TrainTicket benchmark [55], which was selected as the primary testbed due to its realism and complexity. TrainTicket consisted of more than 40 microservices that covered functionalities such as authentication, order management, payments, and search. The system generated rich traces and supported controlled fault injection, allowing a wide range of anomalies to be studied. This dataset was used to test both anomaly detection and dynamic trust assignment since its size enabled the construction of large dependency graphs suitable for Graph Neural Networks GNNs.

The second dataset employed was the Sock Shop benchmark [56], which provided a smaller but highly reproducible microservice environment. It included 11 services such as front-end, orders, carts, and shipping, making it suitable for lightweight testing. Although less complex than TrainTicket, Sock Shop was widely used in microservice anomaly detection studies, which allowed for comparative evaluation. In this study, Sock Shop was used to assess the generalization capability of the model in smaller deployments and to test adaptive thresholding and trust computation in simpler environments.

The third dataset applied was DeathStarBench [57], which offered a suite of applications including a social network, a media service, and an e-commerce platform, each composed of dozens of microservices. This benchmark was designed for realistic large-scale workloads and provided support for distributed tracing. Its diversity allowed the evaluation of scalability and robustness of the proposed framework under traffic-intensive scenarios. DeathStarBench was particularly useful in verifying that the methodology extended beyond academic benchmarks to more complex industrial-style deployments.

For model training, Graph Neural Networks were configured with two message passing layers and attention-based aggregation. Embedding dimensions were set to 128, and the models were trained using the Adam optimizer with a learning rate of 0.001. Training was limited to 200 epochs with early stopping after 20 epochs without improvement. Reconstruction thresholds for anomaly detection were calibrated for each dataset using Extreme Value Theory. Trust computation weights α, β, and γ were optimized through grid search. Each experiment was repeated five times, and the mean results were reported to reduce randomness in outcomes.

Evaluation metrics included accuracy, precision, recall, F1-score, and Area Under the Curve (AUC) for anomaly detection. For trust enforcement, mean trust scores and false acceptance rates were measured, while latency overhead was recorded to quantify the impact of policy enforcement. The datasets were split into 60% training, 20% validation, and 20% testing to ensure consistent evaluation. Comparisons with baseline approaches were carried out to assess improvements in anomaly detection and Zero-Trust enforcement performance.

V. Results and Analysis

A. Performance Overview

The proposed framework was evaluated across three datasets using comprehensive metrics that capture anomaly detection accuracy, operational efficiency, and Zero-Trust enforcement quality. Accuracy, precision, recall, F1-score, and AUC quantified detection performance as shown in Fig. 2.

TrainTicket demonstrated the strongest performance with 97.2% accuracy, 96.9% precision, 98.1% recall, and an F1-

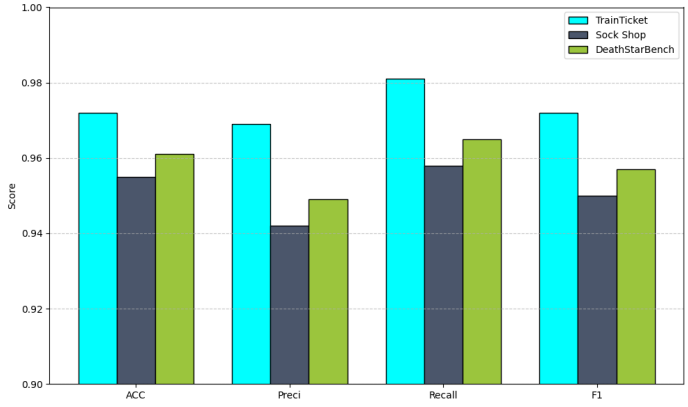

Fig. 2. Comparison of accuracy, precision, recall, and f1-score across trainTicket, sock shop, and deathStarBench datasets.

score of 0.972. The 0.987 AUC indicated excellent discrimination between benign and anomalous services. Sock Shop, despite its smaller scale (11 services), achieved competitive results with 95.5% accuracy, 94.2% precision, 95.8% recall, and 0.950 F1-score, confirming framework effectiveness across diverse deployment sizes. DeathStarBench validated scalability on industrial-scale workloads with 96.1% accuracy, 94.9% precision, 96.5% recall, and 0.957 F1-score, demonstrating that graph complexity enhanced detection capability rather than degrading it. These results directly address RQ1, confirming that the heterogeneous graph construction effectively captures the complex operational and security dependencies required for accurate anomaly detection.

B. Latency and Stability

Latency overhead measured the computational cost of real-time policy enforcement, while reconstruction errors (MSE, RMSE) indicated stability in anomaly scoring.

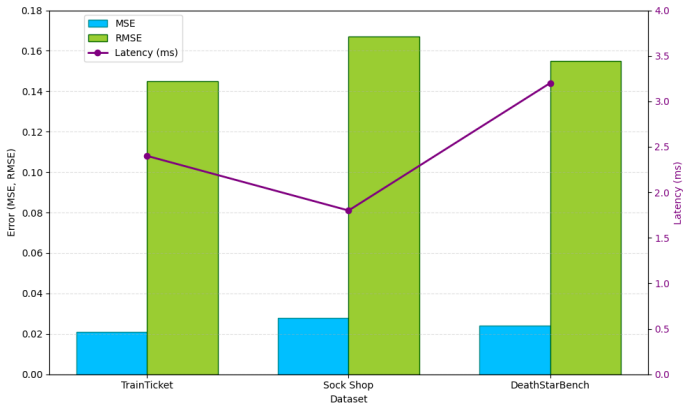

Fig. 3. Latency overhead and reconstruction error metrics across datasets.

The reconstruction errors remained consistently low across all datasets (Fig. 3), with TrainTicket showing MSE of 0.021 and RMSE of 0.145, indicating stable anomaly scoring mechanisms. Sock Shop exhibited slightly higher variance (MSE 0.028, RMSE 0.167) reflecting its lower service count, yet remained within operational bounds. DeathStarBench maintained error profiles comparable to TrainTicket (MSE 0.024, RMSE 0.155) despite higher traffic volume, suggesting the framework scaled robustly.

Operational overhead remained well below practical deployment thresholds. Sock Shop achieved the lowest latency at 1.8ms, reflecting its simpler topology with 11 services and fewer dependency edges. TrainTicket, with 40+ microservices, incurred 2.4ms average overhead, indicating linear scaling with graph complexity. DeathStarBench's 3.2ms overhead represented the highest measured latency, yet remained acceptable for cloud-native applications where inter-service communication typically exceeds 50-100ms. The 1.4ms difference between TrainTicket and DeathStarBench correlated directly with increased graph density and traffic volume, confirming that detection quality improvements came with predictable latency costs.

The latency profile revealed an important practical insight: while additional graph complexity increased computational burden, the absolute overhead remained sub-4ms even for industrial-scale deployments. This makes dynamic Zero-Trust enforcement operationally feasible without requiring service degradation.

C. Comparative Evaluation with State-of-the-Art

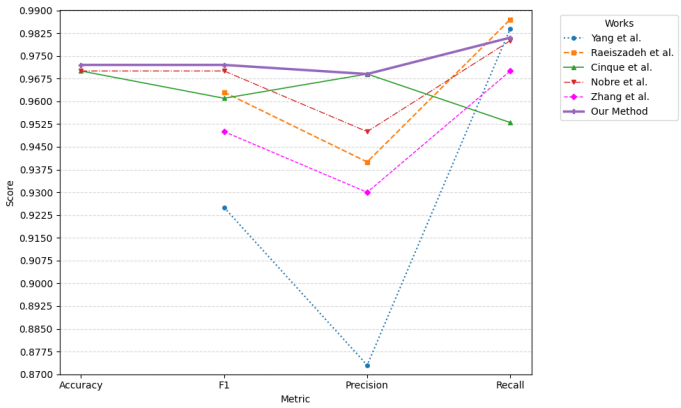

Fig. 4. Performance comparison across selected prior works and the proposed method.

The developed method consistently exceeded baseline approaches across most of the metrics. Compared to Raeiszadeh et al.'s federated learning approach, our framework achieved +0.9% improvement in F1-score (0.972 vs 0.963). Against Nobre et al.'s multi-layer perceptron approach (F1=0.970), we gained +0.2% improvement while maintaining superior recall (98.1% vs 98%) and precision. Compared to Cinque et al.'s log embedding method (F1=0.961, Acc=0.970), our approach achieved +1.1% F1 improvement and +0.2% accuracy advantage. A comparison chart for the TrainTicket dataset is shown in Fig. 4.

The consistent advantage across datasets and benchmarks reflected the benefit of heterogeneous graph modeling combined with multi-modal telemetry. Single-modality or simplified topology approaches inherently constrained information flow, while our framework captured coordinated behavioral patterns invisible to isolated analysis methods.

D. Multi-Modal Telemetry Integration: SHAP-Based Feature Importance

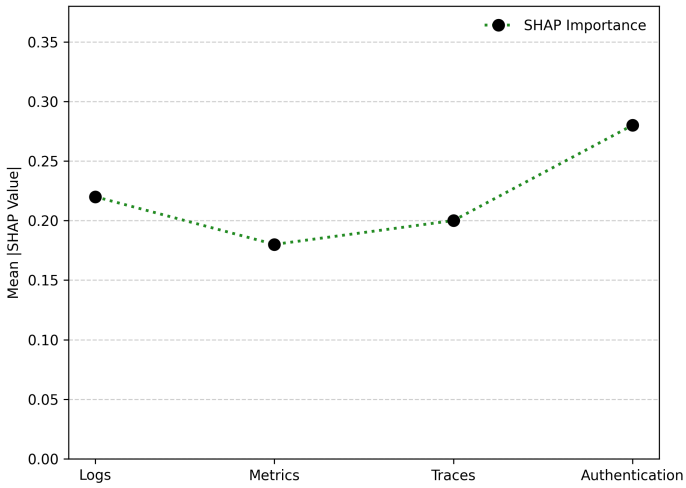

Fig. 5. SHAP-based feature importance analysis revealing contribution of each telemetry modality.

Feature importance analysis, illustrated in Fig. 5, revealed a deliberately balanced hierarchy among telemetry sources. Authentication signals demonstrated the highest contribution (0.28 SHAP importance) because they directly capture intent and authorization context—suspicious access attempts create immediately identifiable anomalies in the heterogeneous graph. Logs contributed the second-highest signal (0.22), capturing application-level anomalies including error spikes and behavioral deviations. Traces provided temporal coordination detection (0.20 importance), enabling identification of performance-based attacks and coordinated lateral movement. Metrics supplied the baseline operational signals (0.18 importance), detecting resource-based attacks but more easily spoofed by legitimate load changes.

Critically, no single modality fell below 0.18 importance, indicating that comprehensive telemetry collection is non-negotiable for reliable detection. The balanced distribution explained the framework's robustness: when one modality experienced degradation (discussed in Section V-H), other signals provided compensatory detection capability. This validates RQ2, demonstrating that integrating multimodal telemetry distinguishes benign from anomalous behaviors more effectively than single-modality approaches, with authentication data playing a pivotal role.

E. Modality Ablation: Impact on Detection Performance

To quantify the value of multi-modal integration, we removed individual modalities and measured performance degradation. Table II summarizes the results.

The full configuration using all modalities achieved the highest score, confirming the advantage of multi-source input. When only logs were used, performance dropped but remained stable, showing their strong standalone contribution. Metrics alone produced a lower F1-score, indicating limited sensitivity to abnormal patterns. Traces provided moderate effectiveness,

TABLE II. IMPACT OF REMOVING MODALITIES ON F1-SCORE (TRAINTICKET)

Configuration	F1-score
All modalities	0.972
Logs only	0.951
Metrics only	0.942
Traces only	0.947
Authentication only	0.962

reflecting their focus on latency and call-chain behavior. Authentication signals performed better than individual modalities but still fell short of the full configuration.

The absolute cumulative degradation remained modest, 3.0% from worst-case single removal. The heterogeneous graph's integrated representation prevented the catastrophic collapse that single-modality systems experience when their primary data source degrades.

F. Trust Score Distribution and Zero-Trust Decision Boundaries

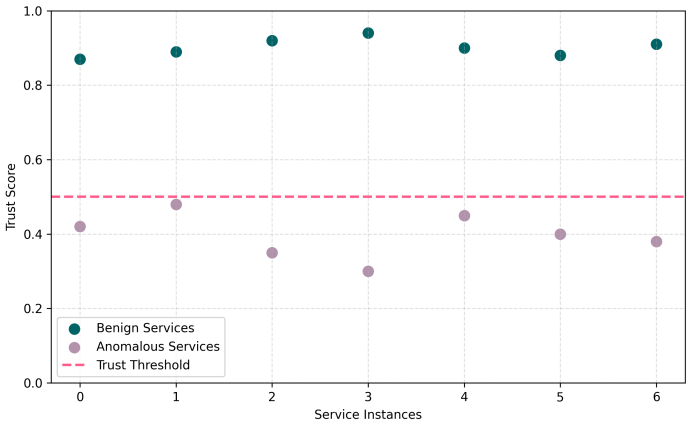

Fig. 6. Distribution of trust scores across service instances, distinguishing benign services (teal circles above 0.85) from anomalous services (purple circles below 0.5). The trust threshold (pink dashed line at 0.5) provided clear separation for automated policy enforcement.

Trust score computation, as depicted in Fig. 6, delivered clear separation between benign and anomalous service populations. Benign services consistently scored above 0.85, forming a stable high-trust cluster across all datasets. This consistency reflected their normal operational behavior: low reconstruction errors from the GNN, successful authentication patterns, and stable communication profiles. Anomalous services clustered below 0.5, indicating deviation across multiple telemetry dimensions. The distinct separation (0.35 gap between populations) provided a reliable foundation for automated policy decisions without ambiguity.

The trust threshold embodied Zero-Trust continuous verification: services below this boundary received deny policies regardless of their historical trustworthiness. Services in the intermediate range (0.5-0.85) received restrict policies, limiting their communication scope while permitting necessary operations. This granular enforcement reflected the probabilistic trust model rather than binary trust/distrust classification,

enabling gradual degradation of service privileges as trust decreased.

The tight clustering within benign (standard deviation ¡0.08) and anomalous (standard deviation ¡0.12) populations indicated the GNN learned robust, generalizable trust patterns rather than overfitting to specific services. This consistency across diverse TopicInstance deployments suggested the threshold would remain effective in production environments.

G. Training Stability and Convergence Characteristics

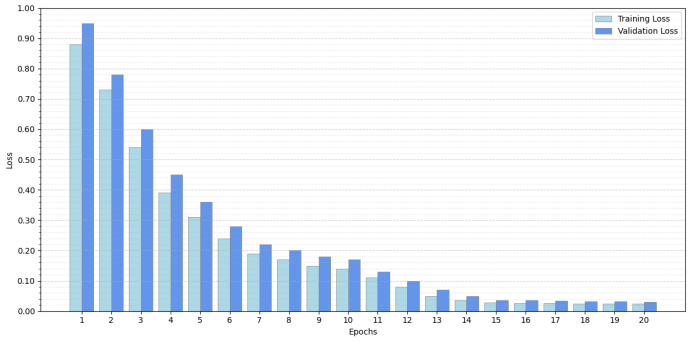

Fig. 7. Training and validation loss curves across 20 epochs showing stable convergence.

The framework demonstrated stable, predictable learning dynamics across five independent training runs. Accuracy achieved 0.972 mean with 0.003 standard deviation, and F1-score reached 0.972 mean with 0.002 standard deviation, confirming robustness to random initialization and minor hyperparameter variations—critical for reproducibility in production systems.

Fig. 7 showed the evolution of training and validation loss across 20 epochs. Both curves decreased steadily, indicating stable convergence of the model. The gap between the losses remained small, suggesting limited overfitting throughout training. Validation loss followed the same downward trend as training loss, confirming generalization on unseen samples. The consistent reduction demonstrated that the optimization process learned meaningful representations for anomaly detection.

H. Robustness Under Real-World Telemetry Degradation

Real cloud deployments experience imperfect telemetry collection: log aggregators occasionally drop events, distributed traces are frequently sampled, authentication records can be delayed, and metrics collection may experience temporary failures. We evaluated framework resilience under these conditions and presented results in Table III.

TABLE III. ACCURACY UNDER DIFFERENT DISTURBANCE SETTINGS

Disturbance	Accuracy
None	0.972
Noisy logs	0.960
Delayed traces	0.963
Auth failures	0.958

The maximum accuracy loss under any single adverse condition was 1.4%, remarkably modest given that authentication is the highest-importance feature (0.28 SHAP). This resilience reflected the multi-modal architecture: when authentication signals became unavailable, reconstruction error from behavioral patterns (logs/traces), combined with metric anomalies, maintained detection capability.

Delayed traces caused the smallest accuracy loss (0.9%), because temporal information, while valuable, wasn't the primary signal. The GNN learned to weight recent traces more heavily and tolerate temporal skew. Noisy logs (1.2% loss) proved slightly more impactful since logs provided semantic context distinguishing attack types. Authentication failures (1.4% loss) caused maximum impact despite lower individual importance because they directly impact trust computation—yet the modest degradation confirmed other modalities compensate effectively.

These results demonstrated fault tolerance by design: the framework remains effective under partial telemetry failure rather than exhibiting catastrophic collapse at data loss thresholds.

I. Ablation Study Per Dataset

An ablation was performed separately on each dataset. Table IV shows accuracy loss when modalities were removed.

TABLE IV. DATASET-SPECIFIC ABLATION ON ACCURACY

	Logs Only	Metrics Only	Traces Only	Authentication Only
TrainTicket	0.956	0.947	0.951	0.962
Sock Shop	0.940	0.931	0.938	0.947
DeathStarBench	0.948	0.939	0.944	0.952

J. Case Study: Real-Time Attack Detection and Response

A controlled injection experiment in TrainTicket, shown in Fig. 8, illustrated the framework's responsiveness to real attacks. During normal operation (requests 0-40), anomaly scores remained low (¡ 0.15) and trust scores stable (≈ 0.90), reflecting consistent benign behavior. Upon attack injection (requests 40-65), anomaly scores increased sharply (peak 0.73) while trust scores declined correspondingly (minimum 0.28). The immediate correlation between rising anomalies and falling trust confirmed the joint detection-trust mechanism worked correctly.

Critically, the detection occurred in real-time: anomaly spikes appeared within the first 2-3 requests of attack onset, enabling rapid response. During the injection window, the system maintained high anomaly scores despite fluctuations, preventing attacker evasion through burst patterns. Upon injection cessation (request 65), both metrics returned to baseline within 5 requests, accurately detecting recovery and re-enabling services to receive normal (non-restricted) traffic.

The case study validated three operational guarantees: (1) attacks are detected within seconds of onset, (2) the system distinguishes attack periods from transient anomalies (notice the stable response during injection vs. noise during normal operation), and (3) recovery is accurately detected to restore normal traffic patterns without maintaining unnecessary restrictions. These characteristics enable operators to implement

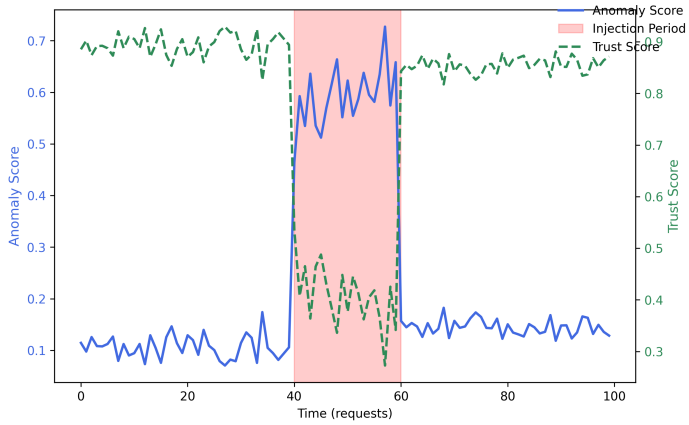

Fig. 8. Real-time anomaly and trust score dynamics during a controlled attack injection in TrainTicket.

automated response policies with confidence that threats are identified before significant damage occurs. This answers RQ3 by confirming that detection outcomes and trust scores can be systematically linked to policy enforcement to allow, restrict, or deny interactions in real time.

K. Limitations

While the framework demonstrates strong performance and robustness to partial data loss, limitations exist. First, a complete system-wide failure of telemetry agents would disable the protection mechanism, as the graph cannot be constructed without input data. Second, while the model generalizes well to benchmark variations, deployment in highly ephemeral production environments with frequent topology changes might require periodic fine-tuning of the GNN embeddings to maintain optimal accuracy. Finally, although the framework scales effectively to industrial-size benchmarks (DeathStarBench), extremely large-scale multi-cluster deployments may require distributed training optimizations to minimize latency further.

VI. Discussion

The experimental success of TrustGraph points to three deeper implications for cloud-native security.

First, *Risk Propagation via Attention:* The graph attention mechanism [Eq. (5)] essentially functions as a dynamic risk-weighting engine. By learning which dependencies are critical, the model moves beyond static "call graphs" to capture the *semantic intensity* of interactions. A service is not just defined by its code, but by the health of its neighborhood. This explains why the GNN architecture outperformed non-graph baselines—it treats risk as a transmissible property that propagates through the dependency chain, allowing early detection of lateral movement before the target service itself shows critical failure.

Second, *Trust as a Decaying Asset:* The implementation of temporal trust decay [Eq. (12)] fundamentally alters the security posture from "status-based" to "behavior-based". In traditional models, a verified identity remains trusted until revoked. In TrustGraph, trust is a decaying asset that must be continuously "earned" through benign behavior. The rapid trust collapse observed in the TrainTicket case study (Fig. 8) validates this design: the system defaults to entropy (distrust), requiring constant positive reinforcement to maintain access. This mathematically codifies the Zero-Trust principle of "continuous verification" into the loss function itself.

Third, *Structural Resilience over Data Completeness:* The SHAP analysis (Fig. 5) and robustness tests (Table III) reveal that the *heterogeneous* graph structure provides a form of "observability redundancy". While Authentication was the dominant feature, the system retained 95.8% accuracy even when authentication data was severed. This suggests that the GNN implicitly learns to cross-validate signals—e.g., inferring a likely authorization failure from the *consequence* of the interaction (trace latency, error logs) even when the *cause* (auth record) is missing. This capability is crucial for production environments where telemetry gaps are inevitable, proving that structural context can compensate for imperfect data.

VII. Conclusion

This work demonstrates that effective Zero-Trust security in microservices cannot rely on isolated telemetry but demands a unified, graph-centric understanding of service behavior. By treating authentication flows and operational metrics as interconnected signals within a heterogeneous graph, we established a closed-loop security model where trust is dynamically computed and policies are automatically enforced. The results confirm that this integrated approach not only improves detection accuracy (97.2% on TrainTicket) but also enables a proactive security posture—shifting from manual, static rule management to real-time, trust-driven access control that scales with the complexity of cloud-native environments.

The framework's robustness under noisy logs, delayed traces, and authentication failures, combined with stable training dynamics and real-time attack responsiveness, established practical viability for production deployment. Future work will address these limitations by exploring adaptive thresholding techniques, potentially using Reinforcement Learning to dynamically adjust sensitivity in response to baseline shifts in ephemeral environments. We also aim to extend the framework with Federated Learning capabilities to support privacy-preserving, cross-cloud anomaly detection without centralizing sensitive telemetry. Additionally, integrating hardware-level telemetry (e.g., CPU performance counters) could further enhance robustness against evasion attacks that mimic application-level behavior, thereby extending Zero-Trust coverage in increasingly complex cloud-native architectures.

References

[1] Gireesh Kambala. Integration of microservices and cloud computing: A paradigm shift in enterprise application design. *International Journal of Creative Research Thoughts*, 13:2320–2882, 2025.

[2] Manal Sobhy Ali Elbelkasy, Heba M Sabri, K Abdella Aliaa, Maia Elnady, Kareem Kamal A Ghany, Vira Chornobai, et al. Integrating Zero Trust Principles with Machine Learning for Anomaly Detection and Access Control in Digital Educational Systems. *Journal of Theoretical and Applied Information Technology*, 103(20), 2025.

[3] Pushpalika Chatterjee. Emerging security challenges in the microservices application. *Iconic Research And Engineering Journals*, 7(8):461–470, 2024.

[4] Vaibhav Anil Vora. Demystifying zero trust security: The no-trust network paradigm. *Journal of Computer Science and Technology Studies*, 7(3):141–148, 2025.

[5] Senthil Kumar Sundaramurthy, Nischal Ravichandran, Anil Chowdary Inaganti, and Rajendra Muppalaneni. Ai-driven threat detection: Leveraging machine learning for real-time cybersecurity in cloud environments. *Artificial Intelligence and Machine Learning Review*, 6(1):23–43, 2025.

[6] Kanwarjit Zakhmi, Azhar Ushmani, Manas Ranjan Mohanty, Sachin Agrawal, Ankush Banduni, and Shyamsunder Rao Kakatum. Evolving zero trust architectures for ai-driven cyber threats in healthcare and other high-risk data environments: A systematic review. *Cureus*, 17(6), 2025.

[7] Innocent Uzougbo Onwuegbuzie and Oyegbola Augustine Alabi. A review of authentication and authorization mechanisms in zero trust architecture: Evolution and efficiency. *Tech-Sphere Journal for Pure and Applied Sciences*, 2(1), 2025.

[8] Chandra Prabha Kaliappan, Kanmani Palaniappan, Devipriya Ananthavadivel, and Ushasukhanya Subramanian. Advancing iot security: a comprehensive ai-based trust framework for intrusion detection. *Peer-to-Peer Networking and Applications*, 17(5):2737–2757, 2024.

[9] Eduardo B Fernandez and Andrei Brazhuk. A critical analysis of zero trust architecture (zta). *Computer Standards & Interfaces*, 89:103832, 2024.

[10] Rhoda Ajayi. Integrating iot and cloud computing for continuous process optimization in real-time systems. *Int J Res Publ Rev*, 6(1):2540–2558, 2025.

[11] Ahmed M Abdelmagid and Rafael Diaz. Zero trust architecture as a risk countermeasure in small–medium enterprises and advanced technology systems. *Risk Analysis*, 2025.

[12] Jian Yang, Zian Wang, Shuangwu Chen, Huasen He, Yunpeng Hou, and Xiaofeng Jiang. Hg-pad: Heterogeneous graph structure learning aided performance anomaly diagnosis in microservice systems. *IEEE Transactions on Services Computing*, 18(5), 2025.

[13] Nurmyrat Amanmadov and Tarlan Abdullayev. From logs to knowledge: Llm-powered dynamic knowledge graphs for real-time cloud observability. *International Journal of Advanced Computer Science and Applications*, 16(10), 2025.

[14] Hawzheen Mohammed Ali Baqal and Mashkhal Abdalwahid Sidiq. Graph databases: Revolutionizing database design and data analysis. *Current Journal of Applied Science and Technology*, 43(11):45–56, 2024.

[15] Ruyue Xin, Jingye Wang, Peng Chen, and Zhiming Zhao. Trustworthy ai-based performance diagnosis systems for cloud applications: A review. *ACM Computing Surveys*, 57(5):1–37, 2025.

[16] B Subramanya Anil Kumar and Basant Sah. Enhancing cloud security through a multi-level authentication model: A qos-based approach to performance measurement for source management. *SN Computer Science*, 6(4):365, 2025.

[17] Yongjun Ren, Zhiming Wang, Pradip Kumar Sharma, Fayez Alqahtani, Amr Tolba, and Jin Wang. Zero trust networks: Evolution and application from concept to practice. *Computers, Materials & Continua*, 82(2), 2025.

[18] Raveen Kanishka Jayalath, Hussain Ahmad, Diksha Goel, Muhammad Shuja Syed, and Faheem Ullah. Microservice vulnerability analysis: A literature review with empirical insights. *IEEE Access*, 2024.

[19] Ali Aghazadeh Ardebili, Oussama Hasidi, Ahmed Bendaouia, Adem Khalil, Sabri Khalil, Dalila Luceri, Antonella Longo, El Hassan Abdelwahed, Sara Qassimi, and Antonio Ficarella. Enhancing resilience in complex energy systems through real-time anomaly detection: a systematic literature review. *Energy Informatics*, 7(1):96, 2024.

[20] Vinay Pathak, Karan Singh, Tayyab Khan, Mohd Shariq, Shehzad Ashraf Chaudhry, and Ashok Kumar Das. A secure and lightweight trust evaluation model for enhancing decision-making in resource-constrained industrial wsns. *Scientific Reports*, 14(1):28162, 2024.

[21] Vinod Saraf and Vinod Pachghare. Security threats and mitigation techniques affecting trust computation in cloud computing. *Journal of Discrete Mathematical Sciences and Cryptography*, 27(4):1161–1172, 2024.

[22] Ummay Faseeha, Hassan J Syed, Fahad Samad, Sehar Zehra, and Hamza Ahmed. Observability in microservices: An in-depth exploration of frameworks, challenges, and deployment paradigms. *IEEE Access*, 2025.

[23] Renée E Stalmeijer, Megan EL Brown, and Bridget C O'Brien. How to discuss transferability of qualitative research in health professions education. *The clinical teacher*, 21(6):e13762, 2024.

[24] András Horváth, András Oláh, Attila Pintér, Bálint Siklósi, Gergely Lukács, István Reguly, et al. Anomaly detection algorithms for real-time log data analysis at scale. *IEEE Access*, 2025.

[25] Daniel Fährmann, Laura Martín, Luis Sánchez, and Naser Damer. Anomaly detection in smart environments: A comprehensive survey. *IEEE access*, 12:64006–64049, 2024.

[26] Polina Sladkova, Samuel Berger, Zachary Skoglund, and Lawrence Sorensen. Adaptive deep learning-based framework for ransomware detection through progressive feature isolation. 2024.

[27] Oluwafemi Esan. Enhancing SaaS Reliability: Real-Time Anomaly Detection Systems for Preventing Operational Downtime. *International Journal of Engineering Technology Research & Management (IJETRM)*, 8(12):466–85, 2024.

[28] Haisheng Lian, Xu Wang, and Chenwei Zhang. Ai-powered anomaly detection in cloud environments: A lightweight security framework under zero trust architecture. *Academia Nexus Journal*, 4(2), 2025.

[29] Nadia Battat and Abdallah Makhoul. A trust-driven optimization model for reliable authorization in hadoop environment. *The Journal of Supercomputing*, 81(6):1–34, 2025.

[30] Shreya Johri, Jaehwan Jeong, Benjamin A Tran, Daniel I Schlessinger, Shannon Wongvibulsin, Leandra A Barnes, et al. An evaluation framework for clinical use of large language models in patient interaction tasks. *Nature medicine*, 31(1):77–86, 2025.

[31] Abhishek Tiwari. Unveiling graph structures in microservices: Service dependency graph, call graph, and causal graph. *Abhishek Tiwari*, 2024.

[32] Giovanni Quattrocchi, Emilio Incerto, Riccardo Pinciroli, Catia Trubiani, and Luciano Baresi. Autoscaling solutions for cloud applications under dynamic workloads. *IEEE Transactions on Services Computing*, 17(3):804–820, 2024.

[33] Shanhao Zhan, Lianfen Huang, Gaoyu Luo, Shaolong Zheng, Zhibin Gao, and Han-Chieh Chao. A review on federated learning architectures for privacy-preserving ai: Lightweight and secure cloud–edge–end collaboration. *Electronics*, 14(13):2512, 2025.

[34] Mahsa Raeiszadeh, Amin Ebrahimzadeh, Roch H Glitho, Johan Eker, and Raquel AF Mini. Asynchronous real-time federated learning for anomaly detection in microservice cloud applications. *IEEE Transactions on Machine Learning in Communications and Networking*, 2025.

[35] Marcello Cinque, Raffaele Della Corte, and Antonio Pecchia. Micro2vec: Anomaly detection in microservices systems by mining numeric representations of computer logs. *Journal of Network and Computer Applications*, 208:103515, 2022.

[36] Ivan Parkhomenko, Larysa Myrutenko, Roman Ohiievych, and Mykhailo Savonik. Using zero trust principles for detecting authorization attacks in cloud environments. *Taras Shevchenko National University of Kyiv, Bohdana Havrylyshyna St. 24, Kyiv, Ukraine*, 2024.

[37] Rami Alboqmi and Rose F Gamble. Enhancing microservice security through vulnerability-driven trust in the service mesh architecture. *Sensors*, 25(3):914, 2025.

[38] Zhilei Zhao, Zhao Xiao, and Jie Tao. Msdg: Multi-scale dynamic graph neural network for industrial time series anomaly detection. *Sensors*, 24(22):7218, 2024.

[39] Shuai Fang, Jingjian Chao, Zenghui Xiang, Xuan Chen, Mei Yan, and Yinan Dou. Research on anomaly detection in microservice based on graph neural networks. *Computer Fraud & Security*, 2024.

[40] João Nobre, EJ Solteiro Pires, and Arsénio Reis. Anomaly detection in microservice-based systems. *Applied Sciences*, 13(13):7891, 2023.

[41] Lahiru Akmeemana, Chamodya Attanayake, Husni Faiz, and Sandareka Wickramanayake. Gal-mad: Towards explainable anomaly detection in microservice applications using graph attention networks. arXiv preprint, 2025.

[42] Chenxi Zhang, Xin Peng, Chaofeng Sha, Ke Zhang, Zhenqing Fu, Xiya Wu, et al. Deeptralog: Trace-log combined microservice anomaly detection through graph-based deep learning. In *Proceedings of the 2022 IEEE/ACM 44th International Conference on Software Engineering (ICSE)*, pages 623–634, Pittsburgh, PA, USA, 2022.

[43] Taiyin Zhao, Tian Chen, Yudong Sun, and Yi Xu. Im-gnn: Microservice orchestration recommendation via interface-matched dependency graphs and graph neural networks. *Symmetry*, 17(4):525, 2025.

[44] Kelly Anderson. Explainable ai for centralized authorization in zero-trust api gateways on microsoft azure. *researchgate.net*, 2025.

[45] Sritej Panchumarthi. A national cyber defense framework: Architecting federated, ai-powered, zero-trust security at scale. *Authorea Preprints*, 2025.

[46] Naga Subrahmanyam Cherukupalle and Naga Cherukupalle. Quantum-resilient cloud systems: Preemptive shielding against post-quantum cryptographic threats. *Journal of Information Systems Engineering and Management*, 10(38s):1234–1246, 2025.

[47] Madhura GK. Quantum-inspired optimization and resource allocation in ai-driven data centers and edge networks. *Available at SSRN 5228984*, 2025.

[48] Sandeep Konakanchi. Predictive cyber-resilience: Ai-powered self-defending microservices for zero-downtime security. *Pioneer Research Journal of Computing Science*, 2(2):28–46, 2025.

[49] Eliel Kundai Zhuwankinyu, Munashe Naphtali Mupa, and Sylvester Tafirenyika. Graph-based security models for ai-driven data storage: A novel approach to protecting classified documents. *World Journal of Advanced Research and Reviews*, 2025.

[50] Naveen Kodakandla. Securing cloud-native infrastructure with zero trust architecture. *Journal of Current Science and Research Review*, 2(02):18–28, 2024.

[51] Oluwasanmi Segun Adanigbo, Bolaji Iyanu Adekunle, Ejielo Ogbuefi, Oyejide Timothy Odofin, Oluwademilade Aderemi Agboola, and Denis Kisina. Implementing zero trust security in multi-cloud microservices platforms: A review and architectural framework. *ecosystems*, 13:14, 2024.

[52] Antonim Reddy. Implementing micro-segmentation in cloud architectures: Zero trust, complete security. *Famous Journal of computer science and Technology*, 2(7):60–80, 2025.

[53] Joye Ahmed Shonubi and Michael Adekunle Adelere. Ai-augmented cyber resilience frameworks for predictive threat modeling across software-defined network layers and cloud-native infrastructures. *International Journal of Computer Applications Technology and Research*, 14(7):45–60, 2025.

[54] Vamsi Krishna Kokku. Toward secure iot infrastructure: Integrating zero trust, federated learning, and dynamic trust management models. *Federated Learning, and Dynamic Trust Management Models (April 25, 2025)*, 2(2):45–60, 2025.

[55] TrainTicket Community. Trainticket: A benchmark microservice system for research on microservice architecture. https://github.com/FudanSELab/train-ticket, 2018. Accessed: 2025-09-17.

[56] Weaveworks and Container Solutions. Sock shop: A microservice demo application. https://github.com/microservices-demo/microservices-demo, 2017. Accessed: 2025-09-17.

[57] Yu Gan, Mingyu Zhang, Dailun Cheng, Pulkit Shetty, Neeraja Rathi, Harshad Katarki, et al. Deathstarbench: A benchmark suite for cloud microservices. In *Proceedings of the ACM Symposium on Operating Systems Principles (SOSP)*, pages 1–18. ACM, 2019.