# Graph Neural Networks for Multi-Entity Access Modeling in Zero Trust Architectures

Filippo Bernabucci[1,†], Gianluca Bonifazi[1,*,†], Andrea Rossi[1,†] and Luca Spalazzi[1,†]

[1]DII, Polytechnic University of Marche, Italy

**Abstract**

In this paper, we introduce a new dataset for analyzing access events in Zero Trust Architectures (ZTA), characterized by a multi-entity relational structure that integrates information from users, devices, network contexts, and application resources. Unlike traditional datasets, in which requests are modeled as independent instances, the proposed dataset explicitly captures the relationships between heterogeneous entities, reflecting the inherently relational nature of access decisions in Zero Trust environments. On this basis, we formalize the problem of classifying suspicious requests as a heterogeneous graph task and propose the use of Graph Neural Networks (GNNs) to learn contextual representations of the entities involved. Through an experimental evaluation, we compare various GNN models with non-relational neural network baselines demonstrating that the graph-based approach leads to substantial improvements in classification performance. The results highlight how the advantage of GNNs comes from their ability to model structural dependencies and relational patterns between entities, which are difficult to capture using traditional tabular models. The work thus contributes both a new benchmark for event classification in Zero Trust contexts and empirical and conceptual evidence regarding the importance of relational inductive bias in this domain.

**Keywords**

Graph Neural Networks, Zero Trust Architecture, Graph Learning, Trust Modeling, Cybersecurity

## 1. Introduction

Security architectures based on the Zero Trust paradigm are emerging as a de facto standard for protecting distributed systems and cloud environments, replacing the traditional perimeter-based model. In a Zero Trust environment, every access request must be dynamically evaluated based on multiple factors, including user identity, device characteristics, network context, and the requested resource. This decision-making process is intrinsically complex and requires the ability to integrate and correlate heterogeneous information from various sources.

In recent years, approaches based on machine learning techniques have been proposed to assist Policy Decision Points (PDPs) in classifying suspicious requests. However, most of these approaches model access requests as independent instances, represented in tabular form, while ignoring the relationships between the entities involved. At the same time, the availability of suitable public datasets for analyzing events in Zero Trust scenarios is extremely limited. Existing datasets in the cybersecurity domain often focus on network traffic or packet-level intrusion detection and do not reflect the multi-entity and multi-source nature of access decisions in Zero Trust environments. In particular, there is a lack of a benchmark that would allow for a systematic study of the role of relationships between entities in the classification of suspicious requests.

In this paper, we address both of these gaps. First, we introduce a new dataset for analyzing access events in Zero Trust environments, constructed by correlating heterogeneous logs collected in a controlled cyber range. Second, we propose a formalization of the problem as a heterogeneous graph classification task, in which users, devices, network contexts, and resources are represented as nodes,

*1st Workshop on Graphs Across AI: From Structural Reasoning to Foundation Models, held in conjunction with the IEEE World Congress on Computational Intelligence (WCCI 2026), June 21–26, 2026, Maastricht, The Netherlands*

*Corresponding author.

†These authors contributed equally.

s1111331@studenti.univpm.it (F. Bernabucci); g.bonifazi@univpm.it (G. Bonifazi); s1104126@studenti.univpm.it (A. Rossi); l.spalazzi@univpm.it (L. Spalazzi)



© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073

published 2026-11-08

CEUR-WS.org/Vol-4319/paper6.pdf

and their interactions as edges. On this representation, we use Graph Neural Networks (GNNs) to learn contextual embeddings of entities, leveraging message-passing mechanisms that allow information to propagate along the graph's relationships.

The main contributions of this work can be summarized as follows: *(i)* we introduce a new dataset for classifying access requests in Zero Trust environments, characterized by a multi-entity and multi-source structure; *(ii)* we propose a heterogeneous graph formalization of the problem, which explicitly captures the relationships between users, devices, networks, and resources; *(iii)* we conduct a comparative study between GNN models and non-relational baselines, highlighting the crucial role of relational inductive bias; *(iv)* we provide empirical evidence demonstrating that graph modeling leads to substantial improvements in the classification performance of suspicious requests.

The rest of this paper is organized as follows. Section 2 discusses related work; Section 3 describes the dataset and the construction process; Section 4 introduces the graph formalization and the models used; Section 5 presents the experimental evaluation; and Section 6 concludes the paper and discusses possible future directions.

## 2. Related Literature

The methodology and the proposed development directions of this work are based on an initial investigation of the related literature. This early analysis has helped to map the evolving landscape of artificial intelligence applications within the Zero-Trust framework, highlighting current trends.

Freed and Jackson (2022) [1] explore the intersection of Zero Trust Architecture and Artificial Intelligence. The work discusses the use of behavioral analytics, anomaly detection and predictive modeling to automate security responses and improve identity verification and endpoint defenses. It also outlines future directions including federated learning, explainable AI and blockchain integration as potential enablers for stronger Zero Trust frameworks. Also Joshi (2025) [2] examines emerging technologies that advance Zero-Trust maturity, emphasizing AI enabled mechanisms for dynamic trust evaluation and adaptive security controls. The study highlights how trust scoring models can integrate contextual, behavioral and risk based signals to support continuous verification and policy enforcement. Machine Learning and intelligent assessment techniques are considered key enablers for responsive access decisions. Ahmadi (2025) [3] proposes an AI driven identity based threat segmentation approach that continuously evaluates risk scores enabling automated threat segmentation and rapid identification of compromised identities. This work highlights the importance of dynamic risk computation in enhancing responsiveness to evolving threats. In addition, Ali and Elaffendi (2025) [4] investigate enhanced incident response strategies to counter Advanced Persistent Threats (APTs). The authors propose a suite of six machine learning-based detection modules that, when applied to extended APT datasets, significantly reduce false positives and improve detection accuracy compared to traditional approaches. This study illustrates that the growing sophistication of APTs demands stronger AI augmented detection mechanisms capable of rapid and automated assessment. Finally, the works of Prajwalasimha et al. (2025) [5] introduces a comprehensive AI-driven perspective on enhancing Zero-Trust security. Among the methods suggested, the potential of graph neural networks is highlighted to capture complex interdependencies among entities. This work, therefore, has directed the research towards the study of GNNs and their application possibilities. From this point, the exploration focused on the existing applications of Graph Neural Networks (GNNs) in the field of cybersecurity. Jmal et al. (2023) [6] propose a transferable graph neural network specifically designed to identify attack paths in network configurations by modeling network assets, vulnerabilities, and interactions as a graph and learning shortest path features indicative of compromise sequences. Charkhabi et al. (2024) [7] define a graph based threat classification approach that constructs a representation of cybersecurity data and applies GNNs to identify attack tactics aligned with the MITRE ATT&CK framework. The graph encodes hosts and interactions as nodes and edges to capture specific attack patterns from normal, more random behavior. The study demonstrates how GNNs can enhance threat classification accuracy by integrating topological context and relational semantics. Tran and Park (2024) [8] introduce a graph

embedding framework tailored for network intrusion detection systems that processes network flow data into graph structures where nodes represent flow features and edges encode IP based relationships. Unlike other GNN approaches that may rely on topological graphs built from hosts or events, this model constructs graphs directly from flow feature correlations.

Finally, the last step of the research focused on the existing implementations of these technologies directly within Zero-Trust architectures, in order to enable a comparison with the idea of the proposed model. Amanmadov et al. (2025) [9] implement a heterogeneous graph neural network framework for dynamic Zero-Trust policy enforcement in cloud microservice environments. The system constructs a heterogeneous graph where nodes represent microservices and edges encode diverse telemetry, allowing the attention based GNN to propagate risk and contextual information across dependencies. The model computes dynamic trust scores for real time policy decisions that restrict, allow, or deny interactions. Madireddy (2025) [10] proposes a dynamic, attention based GNN framework for adaptive threat detection in cloud Identity and Access Management logs by modeling user, role, session, and access events interactions as a heterogeneous dynamic graph. This graph representation captures temporal, relational, and contextual dependencies in IAM activity, enabling the model to learn latent interaction patterns that distinguish evolving malicious behaviors from benign operations. Chen et al. (2026) [11] define an anomaly detection framework that integrates Zero-Trust Architecture with graph neural networks to address data transmission security risks. The approach constructs a k-nearest neighbor graph over multi entity relationships and uses GraphSAGE for neighborhood sampling and message aggregation, amplifying structural differences between normal and anomalous behaviors. Khader et al. (2026) [12] propose a predictive cybersecurity framework that constructs a heterogeneous cybersecurity knowledge graph where nodes represent threat intelligence indicators and edges capture semantic and temporal relationships among them. The model applies multiple GNN architectures, including GCN, GraphSAGE, and GAT, to learn relational patterns that indicate the likelihood of zero day exploitation based on historical and contextual signals. In conclusion, the literature provides valuable insights into the current technological landscape and offers useful ideas for implementation. Nevertheless, the system representation and the tools employed for model development differ from those in the studies reviewed, which supports the validity of the results obtained using the proposed methodology.

## 3. Dataset

To the best of our knowledge, no publicly available dataset captures the multi-layer, correlated access events that a dynamic PDP requires. Existing corpora target packet-level intrusion detection in perimeter-based models, a fundamentally different problem. We therefore designed an ad-hoc cyber range architecture to generate a labeled dataset covering the full ZTA stack, from Layer 2 ARP events to application-level enriched audit records.

The infrastructure runs on Microsoft Hyper-V, partitioned into three isolated security tiers enforced by a dual-homed Next-Generation Firewall (OPNsense + Zenarmor DPI), as shown in Figure 1.

A passive network probe runs four open-source services in promiscuous mode: Snort 2.9 (signature-based IDS), tcpdump (full-packet forensic capture), p0f (passive OS fingerprinting via TCP/IP stack analysis), and ARPwatch (Layer 2 ARP-spoofing detection). All events converge in Splunk Enterprise through a taxonomy of over 60 sourcetypes, each carrying SHA-256 integrity and HMAC-SHA256 authentication signatures. On top of that, the Blazor WebAssembly SPA (.NET 10) is the primary data-collection focal point. Unlike passive sensors, it simultaneously observes identity context (JWT claims, session state), device context (browser fingerprint, X.509 certificate), and behavioral context (request rate, route patterns), making its audit records qualitatively richer than any single network source. Every HTTP request traverses two mandatory middleware interceptors. These middlewares verifies both the JWT signature and the live server-side session, enabling immediate revocation independent of token expiry. Each access event then triggers a six-phase enrichment pipeline that extracts HTTP metadata, parses JWT claims, computes the HMAC-SHA256 of the client-side browser fingerprint (~30 device

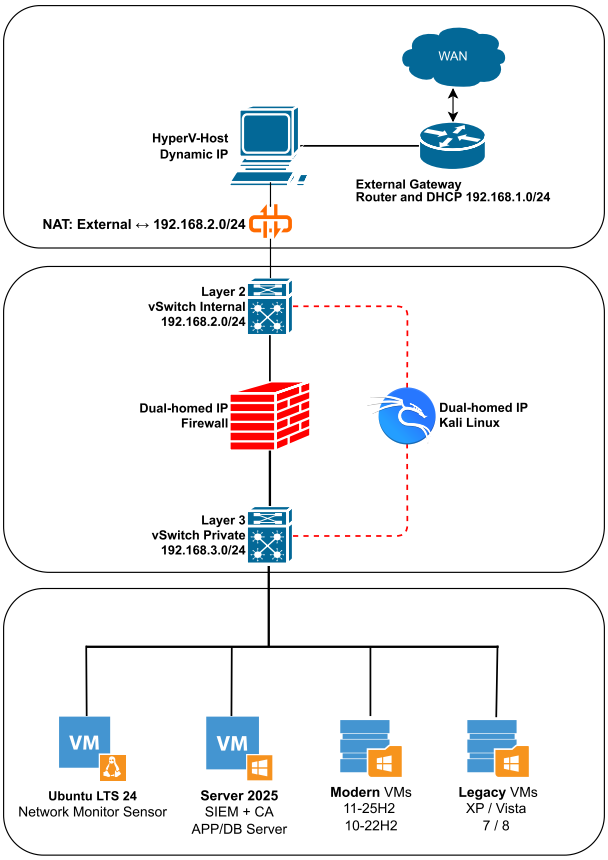



**Figure 1:** Three-tier Zero Trust cyber-range topology

characteristics), detects suspicious activity, and finalises an SHA-256 integrity hash. The resulting audit record is persisted and forwarded synchronously to Splunk, carrying a `Suspicious` boolean label and a categorical description (rate-limit burst, fingerprint mismatch, IP baseline violation, certificate change).

Cross-layer consistency is the primary discriminating feature for trust scoring, i.e., a Linux TCP/IP stack signature observed by `p0f` on an IP whose JWT claims a Windows workstation triggers a trust reduction and automatic session revocation.

## 3.1. Dataset Generation

The labeled dataset is produced by injecting two separated traffic streams into the running infrastructure.

**Legitimate traffic.** A Python bot authenticates against the Blazor application and executes randomised CRUD sequences across all entities, running the client-side fingerprinting module to produce genuine browser-fingerprint headers. All records are labeled `Suspicious = false`.

**Malicious traffic.** A modular penetration-testing suite on a dual-homed Kali Linux node covers the full kill chain: ARP poisoning/flood, TLS downgrade and three-layer MiTM with JSON tampering, JWT manipulation, SQL injection and privilege escalation, and high-volume application fuzzing with honeypot triggering. All records are labeled `Suspicious = true`.

Table 1 highlights how the generated dataset features a complex, multi-source structure, reflecting the typical complexity of Zero Trust scenarios. The variety of entities involved (including users, devices, IP addresses, and resources) confirms the multi-entity nature of the dataset. This balance between different types of events and information sources makes the dataset particularly well-suited for studying

**Table 1**
Dataset composition: event volume and duration per traffic class. All malicious traffics are highlighted in bold.

| Traffic Class | Events | Suspicious | Not Suspicious | Duration |
|---|---|---|---|---|
| CRUD Operations | 111,218 | 12,035 | 99,183 | 6.2 h |
| **Bad Request** | **89,002** | **89,002** | 0 | 4.8 h |
| OPNsense Filter Log | 52,379 | 0 | 52,379 | 6.4 h |
| Chat | 43,072 | 1,613 | 41,459 | 4.5 h |
| **Honeypot** | **12,090** | **12,090** | 0 | 2.0 h |
| **Exception** | **10,804** | **10,804** | 0 | 4.8 h |
| **Alert** | **8,040** | **8,040** | 0 | 2.0 h |
| OPNsense Syslog | 4,888 | 0 | 4,888 | 6.4 h |
| **Rate Limit** | **3,390** | **3,390** | 0 | 2.0 h |
| OPNsense Suricata | 2,635 | 0 | 2,635 | 6.2 h |
| Snort IDS | 2,501 | 0 | 2,501 | 5.1 h |
| Document | 2,010 | 1,981 | 29 | 2.0 h |
| OPNsense NTPd | 1,709 | 0 | 1,709 | 6.4 h |
| Fingerprint | 1,612 | 0 | 1,612 | 5.1 h |
| Authentication | 1,048 | 681 | 367 | 5.9 h |
| **Forbidden** | **995** | **995** | 0 | 5.7 h |
| IIS | 120 | 0 | 120 | 6.2 h |
| Database Backup | 90 | 5 | 85 | 2.0 h |
| OPNsense Lighttpd | 57 | 0 | 57 | 6.3 h |
| OPNsense Unbound | 33 | 0 | 33 | 2.4 h |
| Security Log | 31 | 1 | 30 | 3.3 h |
| OPNsense Captiveportal | 19 | 0 | 19 | 5.1 h |
| Apache Error | 19 | 0 | 19 | 2.3 h |
| OPNsense Dnsmasq | 6 | 0 | 6 | 2.4 h |
| **Total** | **347,768** | **140,637** | **207,131** | |

correlations across different levels of the system, providing a solid foundation for relational modeling approaches such as graph-based ones. The dataset is publicly available on GitHub[1].

As a first step, a set of transformations was applied to the initial and raw dataset in order to extract the fields required for the subsequent processing stages. This step is necessary because the amount of data collected is considerable and heterogeneous, while only a subset of it is relevant for the development of the proposed architecture. In many cases, the same event is recorded by multiple sources, each describing it from a different perspective, resulting in redundant information that can unnecessarily increase the complexity of the analysis. For this reason, a correlation process is performed to link events captured from different sources into a single essential log representing the nature of a request. The transformations applied to the data are designed to generate a list of complete individual requests, enabling them to be processed independently. Events originating from different sources are aggregated based on common parameters, including IP, User, and a restricted time window. This approach makes it possible to group all events performed by a user during a single session in chronological order. As a result, a unified log is generated, containing all the requests issued by a client and enriched with information about the devices present in the same network during the corresponding time interval. Additionally, information related to captive portal events is incorporated. The function ultimately returns a list of these aggregated objects, which can be used to generate the final logs containing individual events. The aim is to produce a simplified format that includes only the essential features required for graph construction and for supporting the computation of trustness weights. Aggregated events are separated into individual logs, containing the corresponding user, IP, and relevant information, including the classification label and the requested resource. Device-related information is linked by matching the extracted IP and timestamp with devices detected on the network at the same time through

[1] https://github.com/andrea-rossi-univpm/ZTA-GNN-GRAAI-Dataset/

fingerprinting. If such information is unavailable, a fallback mechanism searches for the data in captive portal logs or in auxiliary mappings derived from previous IP-MAC associations. If the data is still unavailable, this is due to the request originating from an external network, where device-specific information cannot be retrieved. In such cases, the device is treated as unknown device. Through this correlation process, the quality of the available data is enhanced, enabling more accurate analyses of request structures and potential security incidents. The final structure adopted for the subsequent graph construction and for the experiments is summarized in Table 2. This pre-processed dataset contains 281.692 records, of which 141.123 are legitimate and 140.569 are malicious.

| Field | Type | Description |
|---|---|---|
| user | String | Identifier of the user who generated the event. |
| ip | String | Associated source IP address . |
| role | String | User role at the time of the event; if unavailable, the default value *User* is assigned. |
| mac | String | MAC address of the device associated with the event. |
| hostname | String | Hostname of the device. |
| certificate | String | Client certificate used during the connection.. |
| source | String | Source of the event. |
| event_time | DateTime | Timestamp of the event. |
| event_type | String | Type of the event. |
| resource | String | Resource involved in the event. |
| suspicious | Boolean | Indicator of suspicious behavior; classification label. |
| status_code | Integer | HTTP status code associated with the event. |

**Table 2**
Request final dataset composition.

## 4. Graph Modeling

Starting from the dataset described above, a crucial step consists of extracting the key entities involved in each request and identifying their corresponding trust profiles. This process allows the system to isolate the most relevant information from a large volume of collected data and to represent it in a structured form suitable for graph-based modeling. Each request is abstracted as a communication flow originating from a user and reaching a target resource using an intermediate device and a network infrastructure. Formally,

$$\mathcal{D} = \{(r_i, y_i)\}_{i=1}^{N},$$

where $r_i$ is the $i$-th aggregated request and $y_i \in \{0, 1\}$ is the binary label indicating whether the request is suspicious. Each request is formatted as:

$$r_i = (u_i, d_i, n_i, q_i, x_i, t_i),$$

where $u_i$ identifies the user, $d_i$ the device, $n_i$ the network context, $q_i$ the requested resource, $x_i$ the set of descriptive attributes associated with the event, and $t_i$ the timestamp. In our case, the descriptive attributes include information such as user role, hostname, client certificate, log source, event type, and HTTP status code. The dataset therefore has a multi-entity structure, in which each query does not represent an independent observation, but rather a relationship between heterogeneous entities. From a semantic perspective, each query can be interpreted as a flow of the type: $u_i \to d_i \to n_i \to q_i$.

Starting from the dataset $\mathcal{D}$, we construct a graph representation that explicitly captures the relationships between the entities involved. We define a directed heterogeneous graph as:

$$G = (V, E, \phi, \psi),$$

where:

- $V$ is the set of nodes;
- $E \subseteq V \times V$ is the set of edges;
- $\phi : V \to \mathcal{T}_V$ is a function that associates each node with its type;
- $\psi : E \to \mathcal{T}_E$ is a function that associates each edge with its type.

The set of nodes types is defined as $\mathcal{T}_V = \{\texttt{user}, \texttt{device}, \texttt{network}, \texttt{resource}\}$ while the types of edges are $\mathcal{T}_E = \{\texttt{uses}, \texttt{connects_through}, \texttt{accesses}\}$.

The set of nodes $V$ is obtained as the union of the distinct sets of entities present in the dataset:

$$V = \mathcal{U} \cup \mathcal{M} \cup \mathcal{N} \cup \mathcal{Q},$$

where:

- $\mathcal{U}$ is the set of users;
- $\mathcal{M}$ is the set of devices (identified by their MAC addresses);
- $\mathcal{N}$ is the set of network contexts (IP addresses);
- $\mathcal{Q}$ is the set of resources.

Each node $v \in V$ is associated with a type via the function $\phi$.

For each aggregated request:

$$r_i = (u_i, d_i, n_i, q_i, \ldots) \in \mathcal{D},$$

Let's construct the following arcs: $(u_i, d_i) \in E$ with type $\texttt{uses}$, $(d_i, n_i) \in E$ with type $\texttt{connects_through}$, $(n_i, q_i) \in E$ with type $\texttt{accesses}$. The resulting graph may contain multiple edges between the same pairs of nodes, resulting from different requests. These edges are aggregated into a single weighted edge. In fact, for each pair of nodes $(v_i, v_j)$, we define a weight:

$$w_{ij} = f(c_{ij}),$$

where $c_{ij}$ represents the number of times the relationship between $v_i$ and $v_j$ is observed in the dataset. The weight of the arcs therefore represents the strength of the relationship between two entities.

The complete graph structure is shown in Figure 2.

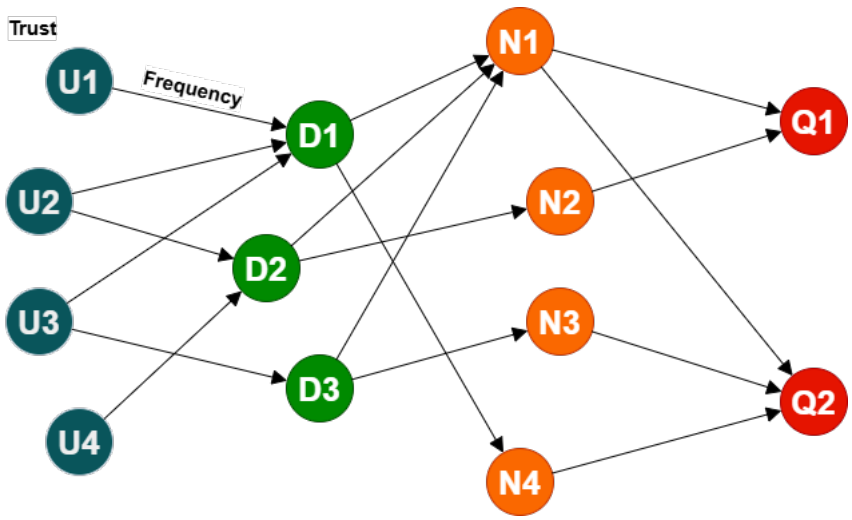



**Figure 2:** Graph diagram.

### 4.1. Trust score

Each node $v \in V$ is associated with an initial feature vector $\mathbf{h}_v^{(0)} = [\mathbf{t}_v \,\|\, S_v]$, where $\mathbf{t}_v$ represents a one-hot encoding of the node type, while $S_v \in \mathbb{R}$ is a *trust score* that models the reliability level of the corresponding entity.

The trust score is a key component of entity representation, as it summarizes historical information regarding behavior observed over time and it constitutes one of the main features that the model uses for its evaluations. Each node is initially assigned a neutral base score $S_{0,v}$, which can depend on additional information extracted from the logs. This value represents an initial estimate of the entity's reliability. For example, the presence of a certificate or a higher hierarchical role can lead to a slight increase, while the absence of essential information results in an initial penalty.

The trust score evolves dynamically based on the outcomes of past requests involving the entity. Let $l \in \{0, 1\}$ be the label associated with a request, where $l = 1$ for suspicious events and $l = 0$ for legitimate events. Each outcome is associated with a score change, negative or positive respectively, denoted by $\delta_s > 0$ and $\delta_c > 0$.

In order to consider the time component, each update is weighted by an exponential decay factor:

$$w_t = e^{-\lambda \Delta t},$$

where $\lambda > 0$ is a parameter that controls the decay rate and $\Delta t$ represents the time elapsed since the event in question. As a result, more recent events have a greater impact than those that occurred further back in time.

The change in trust associated with an event is therefore defined as:

$$\Delta(l, t) = \begin{cases} -\delta_s \cdot w_t, & \text{if } l = 1, \\ \delta_c \cdot w_t, & \text{if } l = 0. \end{cases}$$

The trust score of an entity $v$ at time $t$ is updated incrementally according to the following process:

$$S_v(t) = \begin{cases} S_{0,v} + \Delta(l, t), & \text{if } v \text{ is observed for the first time,} \\ S_v(t-1) + \alpha_v \cdot \Delta(l, t), & \text{otherwise.} \end{cases}$$

The scaling factor $\alpha_v$ allows you to adjust the impact of updates based on the type of entity. In particular, for user-type nodes, the following applies:

$$\alpha_v = S_{0,v},$$

while for all other entities is assumed $\alpha_v = 1$. This choice reflects the assumption that user behavior has a greater impact on risk assessment than other system entities. Furthermore, the introduction of the scaling factor $\alpha_v$, derived from the initial value $S_{0,v}$, allows us to weight the magnitude of trust score updates based on static user characteristics, such as role or the presence of certificates. In this way, malicious actions associated with entities possessing high privileges (e.g., users with administrative roles) result in a more significant change in the score compared to entities with limited privileges, thereby preserving the information incorporated during the trust score initialization phase. It is important to note that the trust score $S_v(t)$ is calculated using only historical information prior to time $t$, thus avoiding any form of information leakage during the training phase. Finally, the value $S_v$ obtained in this way is used as a scalar feature associated with node $v$ in the construction of the graph, contributing to the definition of the representations learned by the GNN.

## 5. Experiments

The proposed model is trained using two main inputs: the PyTorch representation of the graph and a list of encoded requests. This setup enables the GNN to learn node embeddings and use them to

evaluate incoming events. The graph structure is provided as a set of numerical matrix-based elements describing nodes and edges, along with their associated features, including the trust score assigned to each entity. In addition to the graph representation, the model also receives a set of encoded requests that make up the training dataset, so that these patterns can be learned during training. To enable this process, a fixed enumeration of the graph nodes has been defined to uniquely encode graph nodes. This encoding remains consistent as the graph evolves, preserving alignment even when new nodes are added. This allows each request to be traced as a tuple of numerical values compatible with the model, summarizing the event in terms of the involved entities: user, device, network, and resource. The list is paired with a corresponding label array, enabling proper supervision during training and providing a reference for validation.

The GNN updates each node's embedding by message passing, aggregating information from its neighboring nodes based on these inputs, weighted by edge weights and trust scores. This mechanism allows the model to capture both the graph structure and the relationships between nodes. For each quartet, the model retrieves the corresponding node embeddings, as well as the embedded weights of the relations connecting them. The full representation of a request is obtained by concatenating four node embeddings and three edge embeddings. The classification consists of evaluating the subgraph corresponding to the request and is performed by applying a linear combination to the concatenated embeddings, weighted by learned parameters during training. The model produces a probability that the request belongs to class 1.

Binary predictions are obtained by thresholding the predicted probabilities using a fixed value of 0.5 and an adaptive threshold selected by maximizing the F1-score computed from the precision-recall curve on the validation set. The use of both strategies allows for assessing differences in classification performance.

## 5.1. Results

The evaluation was conducted using 5-fold cross-validation, with an 80/20 split between the training and validation sets. The model was trained for 100 epochs using the Adam optimizer, a learning rate of 0.0003, the binary cross-entropy loss function and 64 hidden channels.

The models' performance was evaluated in terms of accuracy and F1-score, considering both a static classification threshold (S.T.) of 0.5 and a dynamic threshold (D.T.) optimized to maximize the F1-score on the validation set. The average results across the 5-fold cross-validation are shown in Table 3.

| **Model** | **Accuracy** | | **F1-Score** | |
|---|---|---|---|---|
| | *S.T.* | *D.T.* | *S.T.* | *D.T.* |
| GCN | 0.9747 | **0.9830** | 0.9748 | 0.9828 |
| GCN without edge weight | 0.9388 | 0.9597 | 0.9413 | 0.9607 |
| GAT | 0.9750 | **0.9830** | 0.9749 | 0.9828 |
| GAT without edge weight | 0.9750 | 0.9829 | 0.9749 | 0.9828 |
| SAGE | 0.9750 | 0.9798 | 0.9749 | 0.9796 |
| SAGE without edge weight | 0.9750 | 0.9813 | 0.9749 | 0.9812 |
| XGBoost | 0.9116 | 0.9646 | 0.9007 | 0.9566 |
| MLP | 0.7693 | 0.8753 | 0.7840 | 0.8575 |
| CNN | 0.7933 | 0.9014 | 0.8016 | 0.8908 |

**Table 3**
Model performance comparison with static and dynamic thresholds.

The results clearly demonstrate the superiority of Graph Neural Network-based models over non-relational baselines. In particular, the GCN, GAT, and GraphSAGE architectures achieve very similar performance, with accuracy and F1-score values close to 0.98 when a dynamic threshold is used. This suggests that the main determining factor is not the specific message-passing architecture, but rather the ability to exploit the relational structure of the graph.

Analysis of variants without edge weights shows that this information contributes significantly to performance, especially in the case of a static threshold. In particular, removing the weights in the GCN convolution layer results in a noticeable drop in performance, while using a dynamic threshold allows for the recovery of most of the performance. This indicates that the arc weights provide a useful but not exclusive contribution, as they are partially compensated for by the decision phase.

As for the specific architectures, GAT (with 4 heads) demonstrates stable and robust performance in both configurations, with minimal variations depending on whether weights are applied to the arcs. GraphSAGE achieves comparable performance, but shows a slight drop in performance with dynamic thresholding in some folds, due to an increase in false positives. However, it stands out for its greater computational efficiency, converging more quickly during training.

Non-relational models perform significantly worse. XGBoost is the most competitive baseline, but it still lags behind GNNs, with a maximum F1-score of approximately 0.95. Traditional neural networks, such as MLPs and CNNs, perform significantly worse, showing marked difficulty in correctly identifying suspected cases and producing a high number of false negatives.

Overall, the results demonstrate that using a graph representation allows for the capture of complex relational patterns among the entities involved, significantly improving classification performance compared to models based on tabular features. Furthermore, the narrow performance gap between the different GNN variants suggests that the main gain stems from the relational inductive bias introduced by graph modeling, rather than from specific architectural choices.

## 6. Conclusion

In this paper, we addressed the problem of classifying access requests in Zero Trust Architectures, highlighting the limitations of traditional approaches based on independent tabular representations. To this end, we have introduced a new multi-source, multi-entity dataset, generated in a controlled cyber range environment, capable of capturing the inherently relational nature of access decisions in Zero Trust contexts.

Based on this, we proposed formalizing the problem as a heterogeneous graph learning task, in which users, devices, network contexts, and resources are modeled as interconnected nodes. The integration of structural information, link weights, and dynamic trust scores enabled the creation of contextual representations of entities, which were subsequently used to classify requests.

The experimental results clearly show that models based on Graph Neural Networks significantly outperform non-relational baselines. In particular, all the GNN architectures considered achieve high and comparable performance, highlighting that the main enabling factor is the relational modeling of the problem rather than the specific architectural variant. This confirms that access decisions in Zero Trust environments cannot be treated as independent observations, but require models capable of capturing structural dependencies between entities.

Despite the promising results, this work has some limitations. In particular, the dataset was generated in a controlled environment and may not capture the full range of variability found in real-world scenarios. The excellent results achieved are probably also influenced by these factors.

Several research directions are identified for future developments. First, integrating temporal graph neural network models would enable the capture of evolutionary dynamics and temporal patterns in interactions. Second, the introduction of explainability mechanisms, in order to make the model's decisions interpretable in critical contexts such as security. Finally, the integration of the model into a real-world Policy Decision Point systems represents a fundamental step toward evaluating its effectiveness in operational scenarios.

Overall, the work demonstrates that adopting a graph-based relational approach represents a promising direction for improving decision systems in Zero Trust Architectures.

## Declaration on Generative AI

The authors have not employed any Generative AI tools.

## References


[1] G. Freed, M. Jackson, Zero Trust Architecture in AI-Driven Cybersecurity: A Machine Learning Perspective, International Journal of Cybersecurity Intelligence & Cybercrime 5 (2022) 67–82.

[2] H. Joshi, Emerging Technologies Driving Zero Trust Maturity Across Industries, IEEE Open Journal of the Computer Society 1 (2025) 1–12.

[3] S. Ahmadi, Autonomous identity-based threat segmentation for zero trust architecture, Cyber Security and Applications 3 (2025) 100106.

[4] G. Ali, S. Shah, M. ElAffendi, Enhancing cybersecurity incident response: AI-driven optimization for strengthened advanced persistent threat detection, Results in Engineering 25 (2025) 104078.

[5] S. Prajwalasimha, A. Pimpalkar, N. Shelke, D. K. J. B. Saini, Zero Trust Architectures Empowered by AI: A Paradigm Shift in Cloud and Edge Cybersecurity, in: Proceedings of the 3rd International Conference on Sustainable Computing and Data Communication Systems (ICSCDS 2025), 2025, pp. 328–335. IEEE.

[6] H. Jmal, F. Hmida, N. Basta, M. Ikram, M. Kaafar, A. Walker, SPGNN-API: A transferable graph neural network for attack paths identification and autonomous mitigation, IEEE Transactions on Information Forensics and Security 19 (2023) 1601–1613.

[7] S. Charkhabi, P. Samimi, S. Bagui, D. Mink, S. Bagui, Node Classification of Network Threats Leveraging Graph-Based Characterizations Using Memgraph, Computers 13 (2024) 171.

[8] D. Tran, M. Park, FN-GNN: A Novel Graph Embedding Approach for Enhancing Graph Neural Networks in Network Intrusion Detection Systems, Applied Sciences 14 (2024) 6932.

[9] N. Amanmadov, J. Iskanderov, T. Abdullayev, TrustGraph: A Heterogeneous GNN for Dynamic Zero-Trust Policy Enforcement in Microservices, International Journal of Advanced Computer Science & Applications 16 (2025) 31.

[10] V. Madireddy, Graph Neural Network Based Adaptive Threat Detection for Cloud Identity and Access Management Logs, arXiv preprint (2025).

[11] Y. Chen, J. Wang, R. Li, Y. Xiong, H. Yang, Research on Data Transmission Anomaly Detection Based on Zero-Trust Architecture and Graph Neural Networks, AIP Advances 16 (2026) 015222.

[12] S. A. Khader, A. A. Ansari, S. S. Ali, Zero-Day Exploit Prediction Using Graph-Based Deep Learning on Vulnerability and Threat Intelligence Data, International Journal of Science and Engineering Applications 15 (2026) 17–26.