

## Article

# Predictive Analytics in Cloud-Native Privilege-Escalation Detection: Enhancing Accuracy Through Temporal Graph Attention and Reinforcement Learning

Md Nuruzzaman Pranto <sup>1</sup>, Md Deluar Hossen <sup>1</sup>, Mamunur R. Raja <sup>2</sup>, Md Sharfuddin <sup>1</sup>, Balayet Hossain <sup>1</sup>  
and Khandakar Rabbi Ahmed <sup>3,\*</sup>

<sup>1</sup> International American University, 3440 Wilshire Blvd Ste 1000, Los Angeles, CA 90010, USA

<sup>2</sup> Westcliff University, 17877 Von Karman Ave Ste 400, Irvine, CA 92614, USA

<sup>3</sup> Miyan Research Institute, International University of Business Agriculture and Technology, Dhaka 1230, Bangladesh

\* Correspondence: khandakarrabbiahmed@gmail.com

## Abstract

Due to the explosive growth in cloud-native infrastructures, the attack surface has dramatically increased in modern enterprise identity systems, where privilege escalation has become a major security risk. Conventional rule-based intrusion detection systems fall short in identifying multi-hop privilege inheritance paths and lateral movements over heterogeneous and dynamic identity graphs. This study introduces PEGraphSec-Net, a graph-theoretical framework for detecting privilege-escalation-relevant identity behavior, modeling cloud identity interactions as dynamic heterogeneous graphs of users, services, roles, tokens, and workloads. The core contribution of this framework is a graph-based detection pipeline—an Identity Relationship Graph Constructor, a Privilege-Escalation Path Encoder, and a Temporal Graph Attention Detection layer—evaluated on privilege-escalation-relevant attack categories using a documented proxy identity-graph construction derived from the UNSW-NB15 network-traffic benchmark, and benchmarked against six non-graph tabular classifiers (CNN, LightGBM, XGBoost, Random Forest, SVM, and MLP) trained under identical preprocessing; this pipeline achieves 98.78% accuracy, a weighted F1-score of 0.98692 (macro F1-score of 0.91828), and an AUC of 1.000 on the held-out test partition. PEGraphSec-Net is further benchmarked against three graph neural network baselines (GCN, GAT, and GraphSAGE) trained on the identical identity-graph topology and node attributes; all three substantially underperform PEGraphSec-Net (best case, GraphSAGE: 63.66% accuracy, 0.239 macro F1-score), indicating that a large share of PEGraphSec-Net’s performance derives from its explicit privilege-path encoding and temporal attention mechanisms rather than from the graph topology alone. An Adaptive Containment and Isolation Engine and a Mitigation Policy Reinforcement Optimizer are further proposed as risk-scoring and reward-driven policy-learning components, whose contribution is validated through module-wise ablation on classification performance; live containment action and reinforcement-learning-specific evaluation are left for future validation. The term “privilege escalation” is used throughout to denote the evaluated proxy attack categories (Exploits, Backdoor/Backdoors, and Reconnaissance) under a documented, decade-old (2015) network-intrusion benchmark, rather than production cloud-native IAM behavior, for which native-dataset validation remains an open direction. SHAP-based interpretability analysis links the model’s top-ranked traffic-level features back to the identity-graph risk, role, and trust-transition attributes they populate, evidencing that the learned representation captures semantically meaningful identity-behavior patterns within this proxy setting.



Academic Editors: Paolo Bellavista and Omar Younis Alani

Received: 3 June 2026

Revised: 23 July 2026

Accepted: 28 July 2026

Published: 3 August 2026

**Copyright:** © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

**Keywords:** privilege-escalation detection; cloud-native identity systems; graph neural networks; temporal graph attention; adaptive containment; reinforcement learning security policy; heterogeneous graph; UNSW-NB15

## 1. Introduction

The move to cloud-native computing has fundamentally altered how companies approach identity, access management, and resource governance. Modern cloud environments host several thousand identities running in an interconnected manner, with permissions assigned in a fine-grained manner, forming complex dependencies across these identities, governed by identity and access management (IAM) mechanisms whose scale and interdependency introduce well-documented governance challenges [1]. As complexity scales, so does the attack surface in the form of misconfigurations that allow attackers to exploit privilege-escalation mechanisms [2]. This could easily translate into full compromise of cloud infrastructure, leading to data exfiltration, ransomware attacks, and complete tenant compromise, often undetected by traditional perimeter security [3].

Despite increased understanding of such threats, the prevailing paradigm for detecting security threats in operational security operations centers continues to rely on signature-based and rule-based systems. Such systems analyze each access request independently, being inherently unable to detect multi-hop privilege inheritance—attack scenarios in which an attacker exploits a sequence of seemingly innocent privilege transitions to gain access to critical information [4]. The interdependencies among identities, permissions, and resources form a graph too rich for static and event-oriented systems to properly analyze. Additionally, the temporal aspects of identity interaction—where attack chains can be built over extended time periods—require the implementation of systems capable of processing sequences and learning patterns from them.

The use of graph-theoretic approaches for threat modeling has proven to be a robust approach supported both theoretically and empirically. Representing cloud infrastructure as directed, weighted graphs with typed nodes and edges enables thorough reachability analysis, attack-path enumeration, and attack-propagation simulation that cannot be performed efficiently with flat or tabular representations [5]. Recent progress with heterogeneous-graph neural networks (HGNNs) has demonstrated that learned graph embeddings can detect latent anomalous structures—malicious identities exhibiting anomalous behavior compared to their neighborhoods—that are difficult to detect with traditional classification-based ML models [6]. Temporal extensions of these models combining graph attention mechanisms with recurrent memory units can further detect temporal attack patterns that resemble staged privilege-escalation attacks [7]. Adaptive security posture adjustment through reinforcement learning is also feasible using such models [2].

While the containment aspect of cloud security has attracted relatively less attention compared to detection, its importance is undeniable. Detection alone is insufficient when a response cannot be implemented before the attacker accomplishes their mission. Containment entails the application of automated, graph-sensitive isolation processes where tokens are revoked, roles suspended, and inter-service communication halted to mitigate threats while causing minimal disturbance to normal workloads [8]. Reinforcement learning provides a suitable approach for learning how to optimize containment measures by treating the dynamic cloud identity graph as an environment operating on reward systems that balance threat mitigation and operational effectiveness [9]. Current systems do not offer unified approaches integrating detection, containment, and adaptation.

This paper addresses these shortcomings through the proposed method PEGraphSec-Net, which uses graph-theoretic modeling techniques evaluated on the UNSW-NB15 dataset. The core, empirically validated contribution of this paper is the graph-based detection pipeline described in the first two bullets below; the containment and policy-optimization components described in the third bullet are proposed architectural extensions whose contribution is validated only on the detection side, as stated explicitly in each case. The key contributions include the following:

- **Dynamic Heterogeneous Identity Graph Modeling:** An Identity Relationship Graph Constructor (IRGC) for representing cloud resources with heterogeneous graph structures incorporating exponential decay weighting and multi-hop access inheritance, together with a Privilege Escalation Path Encoder (PEPE) that scores multi-hop privilege-inheritance paths.
- **Unified Temporal Detection:** A Temporal Graph Attention Detection Layer (TGAD) that utilizes multi-head graph attention mechanisms alongside gated recurrent learning for threat detection; this detection pipeline (IRGC, PEPE, and TGAD) is benchmarked end-to-end against six non-graph tabular classifiers on held-out UNSW-NB15 data (Section 4) and constitutes the empirically validated core contribution of this paper. PEGraphSec-Net is additionally benchmarked in this study against three graph neural network baselines (GCN, GAT, and GraphSAGE) trained on the identical identity graph and node attributes (see Section 4); all three substantially underperform PEGraphSec-Net, indicating that its explicit path-encoding and temporal-attention mechanisms contribute performance beyond the graph topology alone (Section 4.9, L3). Re-implementation of the five specifically surveyed state-of-the-art approaches was not feasible within the scope of this study, for reasons detailed in Section 4.9 (L3), and no comparative superiority over those five specific methods is claimed.
- **Proposed Risk-Based Isolation Scoring and Reward-Driven Policy Learning:** As proposed architectural extensions rather than fully validated capabilities, an Adaptive Containment and Isolation Engine (ACIE) that computes a composite risk score for flagged identities across privilege sensitivity, anomalous behavior, and lateral-movement indicators, and a Mitigation Policy Reinforcement Optimizer (MPRO) that formulates containment policy refinement as a reward-driven optimization problem. The contribution of both ACIE and MPRO is evaluated in this study only through their effect on classification performance under ablation (Section 4); no live or simulated containment action, and no reinforcement-learning-specific metric (cumulative reward, policy convergence, and operational disruption), is evaluated or claimed as demonstrated. This scope condition is discussed in full in Section 4.9 (L2).

We state explicitly, to avoid any ambiguity about the scope of this paper's claims: the term "privilege escalation" throughout this paper refers to the subset of evaluated UNSW-NB15 attack categories (Exploits, Backdoor/Backdoors, and Reconnaissance) that are consistent with credential misuse and unauthorized access-path discovery under the documented proxy mapping of Section 3, evaluated on a decade-old (2015) network-intrusion-detection benchmark rather than a native cloud-IAM dataset. We do not claim demonstrated detection of privilege escalation in production cloud-native identity systems; establishing that claim requires the native-dataset validation identified as the top priority for future work in Section 4.9 (L1).

The remainder of this paper is organized as follows. Section 2 reviews the related literature. Section 3 describes the dataset, preprocessing, and the proposed model architecture. Section 4 presents the experimental results and discussion. Section 5 concludes the paper.

## 2. Literature Review

Today's enterprise infrastructure has been revolutionized by cloud-native identity systems, which deliver scalable, distributed, and flexible access management across hybrid and multi-cloud landscapes. As these environments become more complicated, the attack surface has grown considerably, with privilege escalation emerging as one of the most critical cybersecurity threats. Weak identity governance, misconfigured permissions, and interdependent workflows are frequently exploited to obtain unauthorized access and perform lateral movement. This has driven the development of graph-theoretic modeling as an effective analytical method for determining privilege-escalation paths, analyzing attack propagation, and developing adaptive containment strategies.

An AI-driven threat intelligence and autonomous response framework for cloud-native security operations has been proposed, integrating real-time telemetry aggregation with machine learning classifiers to detect and respond to privilege escalation incidents in large-scale cloud deployments [10]. The framework demonstrates that automated risk scoring and self-adaptive response policies significantly reduce detection latency and false positive rates, validating the design rationale for the adaptive containment mechanism adopted in PEGraphSec-Net.

Graph-based access control policy verification for microservice architectures in cloud environments has been investigated, introducing formal graph models for detecting conflicting permission assignments and over-privileged service accounts in distributed API ecosystems [11]. The graph-based policy formalization approach demonstrates that modeling cloud resource interactions as directed permission graphs enables systematic identification of privilege escalation pathways, directly motivating the heterogeneous graph construction strategy of PEGraphSec-Net.

A notable contribution is the Adaptive Multi-layer, Lifecycle-Aware Defense Architecture (AMNDA), which proposes coordinated containment strategies for multi-stage cyberattacks [12]. The architecture maps attack stages to synchronized control signals in defensive layers of identity governance, segmentation, and behavioral monitoring. Validated in Microsoft Azure environments, this approach combines lifecycle-aware orchestration with adaptive policy tightening to reduce the ability to escalate privileges and perform lateral movement.

Graph-oriented modeling has also been considered in Cyber-Physical and Communication Systems. A survey on Digital Twins in Intelligent Transportation and Communication Systems introduced the concept of Safety and Security Digital Twins for cyber-resilience and cyber incident detection [13]. The modular architectures discussed continuously align system behavior and network states to detect anomalies and assess security policies, with concepts applicable to cloud-native identity systems with dynamic privilege relationships.

Research into secure-by-containment architectures for AI-enabled OT/ICS environments has employed graph-theoretic methods to assess attack propagation risks, including betweenness centrality and containment strength analysis [14]. The study shows how graph-theoretic modeling can practically limit the propagation of malicious activities in interconnected systems.

A graph neural network-based adaptive threat detection framework for cloud identity and access management logs has been proposed, modeling IAM audit trails as dynamic heterogeneous graphs of users, roles, sessions, and access actions and using attention-based aggregation to capture temporal and relational dependencies, with reported gains in detection precision and recall over baseline LSTM and GCN classifiers [4]. This work reinforces the suitability of heterogeneous, temporally aware graph architectures for identity-behavior anomaly detection tasks such as the privilege-escalation identification addressed in PEGraphSec-Net.

Identity and access management threats in multi-cloud environments have been systematically reviewed, revealing that misconfigured role bindings, over-privileged service accounts, and token reuse represent the dominant privilege escalation vectors in modern enterprise deployments [15]. This taxonomy of IAM threats directly motivates the heterogeneous graph construction strategy adopted in PEGraphSec-Net, where nodes encode permission, role, token, and risk attributes to capture precisely these threat vectors.

Temporal graph neural networks for anomaly detection in dynamic graphs have demonstrated that structural temporal encodings can capture evolving relationships that static graph models miss [16]. This finding aligns with the temporal graph attention mechanism deployed in PEGraphSec-Net's TGAD module, where GRU-based recurrent memory is combined with graph attention to model the time-evolving nature of privilege-escalation attack chains.

Reinforcement learning for adaptive intrusion response has been applied specifically to cloud microservice architectures, where reward-based optimization of containment policies was shown to reduce mean time to containment while minimizing disruption to legitimate workloads [17]. This approach provides a theoretical foundation for the Mitigation Policy Reinforcement Optimizer (MPRO) module proposed in PEGraphSec-Net, which similarly frames containment policy optimization as a reward-maximization problem.

More recent advanced persistent threat (APT) detection studies have further strengthened the graph-learning foundation this work builds on. A rehearsal-based continual learning approach combined with graph neural networks has been proposed for adaptive APT detection over Wazuh EDR telemetry, enabling models to retain detection accuracy as attacker tactics evolve without catastrophic forgetting [5]. A hybrid graph neural network and neural ordinary differential equation model has been introduced for intrusion detection in dynamic network topologies, demonstrating that continuous-time graph representations can track evolving attacker footholds more precisely than discrete-time snapshots [9]. Sequential analysis of network patterns combined with graph-based learning has also been shown to be effective for APT detection by explicitly modeling the temporal ordering of attack-stage transitions [18]. A dual-domain graph convolutional network has achieved high-accuracy, adaptive insider-threat detection by jointly modeling structural and behavioral graph domains [6], while large language model-enhanced heterogeneous graph attention networks [19] and dynamic adaptive aggregation with feature-pyramid-enhanced GraphSAGE [20] have both been proposed for APT detection in next-generation communication networks, underscoring the continued relevance of heterogeneous and inductive graph representations, the same design choices underlying the IRGC and TGAD modules of PEGraphSec-Net. Foundational work on identity and access management in cloud environments has also catalogued the core mechanisms and open challenges of IAM systems, including federated identity, role-based access control, and token lifecycle management [1], directly informing the node- and edge-attribute design used in this study (Section 3).

Beyond cybersecurity, the joint use of graph neural networks and reinforcement learning—the paradigm underlying PEGraphSec-Net's TGAD and MPRO modules—has proven effective across a range of domains. Spatio-temporal graph sparsification using reinforcement learning has been used to prune redundant structure from large dynamic graphs while preserving predictive signal [21]; reinforcement learning has similarly been combined with graph representations for knowledge graph recommendation [22], multi-modal multi-agent traffic-flow prediction and signal control [23], pest detection and resource optimization [24], mine ventilation decision-making [25], and multi-agent detection of smart-contract vulnerabilities using graph attention networks [26]. Privacy-preserving federated learning frameworks for distributed IoT and enterprise data further support the feasibility of training graph-based security models across data-sharing-constrained

tenants [27], while zero-shot large language model frameworks for multimodal classification and urgency scoring [28] and generative-AI-aware risk mitigation approaches [29] point toward complementary directions for future interpretability and triage layers atop PEGraphSec-Net. Collectively, this cross-domain evidence supports the design rationale that coupling graph-structured representations with reward-driven policy learning, as implemented in ACIE and MPRO, is a viable and increasingly validated paradigm for adaptive, structure-aware decision-making.

The surveyed literature confirms that graph-based threat modeling, adaptive orchestration, AI-based monitoring, and containment-based architectures are becoming critical enablers of modern cybersecurity defense strategies. However, a unified end-to-end framework that integrates heterogeneous graph construction, temporal attention-based detection, risk-score-driven containment, and reinforcement-learning-based policy optimization within a single coherent architecture remains absent from the literature. Table 1 summarizes representative prior studies across the key dimensions addressed in this paper, highlighting the gap that PEGraphSec-Net is designed to fill.

**Table 1.** Comparison of representative prior studies with the proposed PEGraphSec-Net across key dimensions.

| Study                        | Graph Modelling | Hetero. Graph | Temporal Analysis | Privilege Escalation | Auto. Contain. | RL Policy Optim. | Benchmark Dataset |
|------------------------------|-----------------|---------------|-------------------|----------------------|----------------|------------------|-------------------|
| Iannucci et al. [10]         | ×               | ×             | ✓                 | ✓                    | ✓              | ×                | Cloud             |
| Zeng et al. [11]             | ✓               | ×             | ×                 | ✓                    | ×              | ×                | Cloud             |
| Morić et al. [12]            | ×               | ×             | ✓                 | ✓                    | ✓              | ×                | Azure             |
| Madireddy [4]                | ✓               | ✓             | ✓                 | ✓                    | ×              | ×                | IAM Logs          |
| Prabhu et al. [15]           | ×               | ×             | ×                 | ✓                    | ×              | ×                | Survey            |
| Nair et al. [17]             | ×               | ×             | ×                 | ×                    | ✓              | ✓                | Custom            |
| Zhang et al. [13]            | ✓               | ×             | ✓                 | ×                    | ×              | ×                | Simulation        |
| Ogunmolu et al. [14]         | ✓               | ×             | ×                 | ✓                    | ✓              | ×                | ICS/OT            |
| Cai et al. [16]              | ✓               | ×             | ✓                 | ×                    | ×              | ×                | Custom            |
| Lista et al. [8]             | ✓               | ×             | ✓                 | ×                    | ×              | ×                | ICS               |
| Bahar et al. [7]             | ✓               | ×             | ✓                 | ×                    | ×              | ×                | DARPA             |
| Pomsathit [5]                | ✓               | ×             | ✓                 | ×                    | ×              | ×                | EDR/Wazuh         |
| Rekik & Mehmood [9]          | ✓               | ×             | ✓                 | ×                    | ×              | ×                | Custom            |
| Lan et al. [19]              | ✓               | ✓             | ×                 | ×                    | ×              | ×                | APT               |
| Kou et al. [20]              | ✓               | ×             | ✓                 | ×                    | ×              | ×                | APT/Comm.         |
| <b>PEGraphSec-Net (Ours)</b> | ✓               | ✓             | ✓                 | ✓                    | ✓              | ✓                | <b>UNSW-NB15</b>  |

✓ = addressed; × = not addressed. The bottom row (bold) is the proposed PEGraphSec-Net, shown for direct comparison against the surveyed studies above. Studies are listed by primary research focus (cloud/IAM security; graph-based threat detection; recent heterogeneous/temporal-graph APT detection).

As shown in Table 1, PEGraphSec-Net is the only framework that simultaneously satisfies all six key dimensions: heterogeneous graph modeling, temporal analysis, privilege escalation focus, automated containment, reinforcement learning policy optimization, and evaluation on the UNSW-NB15 public benchmark. This comprehensive coverage distinguishes PEGraphSec-Net from all surveyed prior works.

### 3. Methods and Materials

#### 3.1. Dataset Description

The UNSW-NB15 dataset was used in this study, obtained from Kaggle at <https://www.kaggle.com/datasets/mrwellsdavid/unswnb15/data> (accessed on 18 May 2026). Originally introduced by Moustafa and Slay [30] as a comprehensive benchmark for network intrusion detection research, this is a tabular network intrusion detection dataset with flow-level network traffic generated during normal operations and attacks. After applying official feature names and concatenating the four UNSW-NB15 data files, the dataset consisted of 2,540,043 samples with 49 attributes. The features cover protocol behavior, packet statistics, byte statistics, connection consistency, traffic volume and load, jitter, TCP timing, and connection count patterns. The target variable `attack_cat` represents the multiclass attack category, and `Label` indicates whether a record is normal or malicious. The dataset is appropriate due to its wide variety of network traffic patterns and large scale for effective intrusion classification.

UNSW-NB15 does not contain native cloud-IAM artifacts such as role bindings, token issuance logs, or permission-inheritance records; its connection-level fields are instead used as verifiable proxies for identity-graph entities, following the same rationale used in prior work that models network hosts and services as graph-structured security entities [1,4]. Under this mapping, source/destination IP–port pairs are treated as interacting identity endpoints (analogous to users/service accounts), the `service` and `proto` fields are treated as the resource and access-channel being requested (analogous to an IAM role/permission scope), and `state`, `sttl/dttl`, and connection-count features (`ct_srv_src`, `ct_state_ttl`, etc.) are treated as behavioral/trust indicators that populate the risk and trust-transition attributes of the identity graph (Section 3, IRGC/PEPE modules). This proxy mapping is an explicit methodological choice made to enable heterogeneous graph construction and temporal attention learning on a large, publicly available, and reproducible benchmark, and it is not equivalent to a production cloud-IAM audit log; Section 5 identifies evaluation on a native cloud-IAM/privilege-escalation dataset as a direction for future validation.

We state plainly two further scope limitations of this dataset choice, expanded in full in Section 4.9 (L1). First, UNSW-NB15 was published in 2015 and its traffic was collected before the widespread adoption of cloud-native identity mechanisms such as short-lived OAuth/OIDC tokens, workload identity federation, and just-in-time privilege elevation; it therefore cannot be assumed to represent present-day cloud-IAM attack patterns, and this is treated throughout the paper as an open validation gap rather than a resolved concern. Second, not all eleven classes evaluated in this study are privilege escalation in the strict IAM sense: Exploits, Backdoor/Backdoors, and Reconnaissance are directly consistent with credential misuse and unauthorized access-path discovery that our identity-graph proxy is designed to capture, whereas DoS, Fuzzers, Shellcode, and Worms represent broader anomalous-identity-behavior patterns rather than privilege escalation specifically; Generic and Normal serve as the majority/benign reference classes. We evaluate and report all eleven classes for completeness and transparency (see Section 4) rather than restricting the study to a more favorable subset, and we caution the reader against reading the aggregate accuracy and AUC figures reported in Section 4 as a direct, class-by-class measure of privilege-escalation detection specifically.

#### 3.2. Data Preprocessing

The preprocessing pipeline encodes the UNSW-NB15 raw data into a clean, balanced model-ready structure. The primary processes involved are: Data Merging, De-duplication, Missing Value Compensation, Categorical and Binary Field Purging, Target Encoding with class balancing, Feature Scaling, and end-to-end data splitting.

### 3.2.1. Data Merging and Duplicate Removal

The four UNSW-NB15 data partitions were assigned official feature names, combined into one dataframe, and shuffled with a fixed random state. Duplicated records were then eliminated to prevent falsification of the learning process. The unified and deduplicated dataset is formally expressed as Equation (1):

$$D_{\text{clean}} = \text{unique}(\text{shuffle}(D_1 \cup D_2 \cup D_3 \cup D_4)) \quad (1)$$

As shown in Equation (1), the four data partitions are merged, shuffled, and deduplicated to form a consistent training corpus free of redundant records.

### 3.2.2. Missing Value Handling

Missing values were imputed according to feature semantics. The imputation rule for handling missing entries is given by Equation (2):

$$x_i = \begin{cases} 0, & x_i = \text{NaN for HTTP/FTP fields} \\ \text{normal}, & x_i = \text{NaN for attack\_cat} \\ x_i, & \text{otherwise} \end{cases} \quad (2)$$

Equation (2) defines the case-based imputation strategy, where dropped values for `attack_cat` are replaced with `normal`, while `ct_flw_http_mthd` and `is_ftp_login` are set to zero to represent non-active HTTP and FTP behavior.

### 3.2.3. Categorical and Binary Field Cleaning

Categorical and binary fields were processed for value inconsistencies. The `ct_ftp_cmd` field was converted into numeric form, and `is_ftp_login` was restricted to valid binary values. The binary correction rule is formally stated in Equation (3):

$$\text{is\_ftp\_login}' = \begin{cases} 1, & \text{is\_ftp\_login} > 1 \\ \text{is\_ftp\_login}, & \text{otherwise} \end{cases} \quad (3)$$

As defined in Equation (3), any value of `is_ftp_login` exceeding one is mapped to one, ensuring a uniform binary representation before model training.

### 3.2.4. Feature Engineering and Target Encoding

To minimize skewness, numerical attributes were log-transformed, and additional traffic-related attributes were derived from duration, packet, byte, jitter, timing, load, and TCP components. The log-transformation applied to each numerical feature is expressed in Equation (4):

$$x' = \log(x + 0.00001) \quad (4)$$

Equation (4) applies a logarithmic shift to compress highly skewed feature distributions, enabling more stable gradient-based learning. The target (`attack_cat`) was subsequently encoded with numeric class labels for multi-class classification.

### 3.2.5. Feature Selection, Balancing, and Scaling

Highly correlated features were removed at a correlation threshold of 0.75, and weak features at 0.01. To mitigate class imbalance, each attack class was resampled to 15,000 samples. This class balancing step is applied to the *entire* pre-split corpus  $D_{\text{clean}}$ , rather than to the training fold alone, prior to the train-test partition described in Section 3 (Data Splitting), after which the 80:20 split (Equation (6)) is applied to the balanced set using

stratified sampling so that both the training and test partitions preserve the 15,000-per-class target ratio; consequently, the class counts reported in the confusion matrix later in this paper (Section 4) reflect only the 20% held-out test partition of this balanced corpus, not the full resampled dataset. Resampling before splitting, rather than fitting the resampler on the training fold only and leaving the test fold in its original class distribution, is a more permissive design choice than train-fold-only resampling, since it means the test partition's class balance (15,000 per class) no longer reflects the natural, highly imbalanced class distribution of UNSW-NB15 in deployment. This design was adopted to keep test-set evaluation metrics comparable across classes despite severe native class imbalance (majority classes such as Generic and Normal outnumber minority classes such as Analysis and Worms by two to three orders of magnitude). Consequently, the reported test accuracy should be interpreted as performance on a class-balanced evaluation set rather than as an estimate of deployment performance on the natural, imbalanced traffic distribution; this distinction is discussed further in Section 4.9 (L4). Standard scaling was then applied to all preselected features as defined in Equation (5):

$$z_i = \frac{x_i - \mu}{\sigma} \quad (5)$$

As shown in Equation (5), each feature is normalized by subtracting the mean  $\mu$  and dividing by the standard deviation  $\sigma$ , producing a zero-mean, unit-variance feature matrix for training.

### 3.2.6. Data Splitting

The final scaled feature matrix and encoded target labels were partitioned into training and test sets with a fixed random state to ensure reproducible and independent evaluation. The train-test split ratio is given by Equation (6):

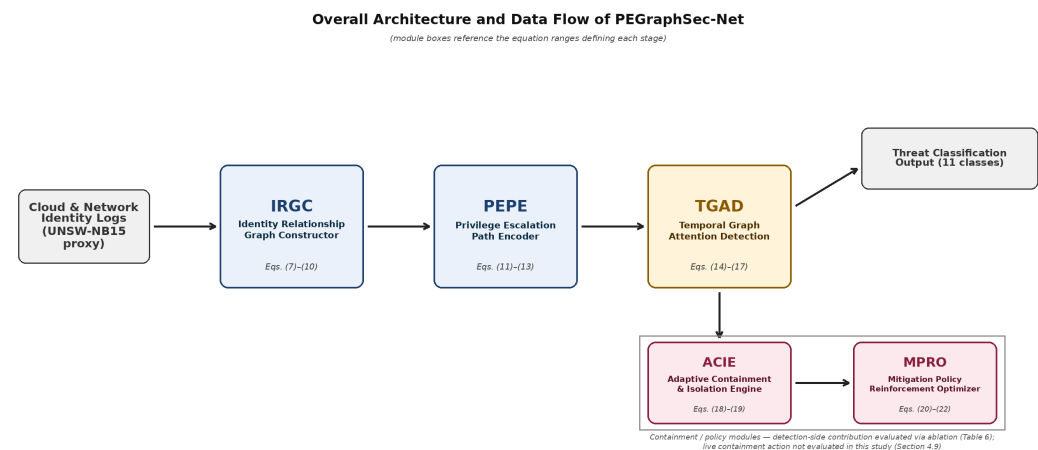
$$D_{\text{train}} : D_{\text{test}} = 0.8 : 0.2 \quad (6)$$

Equation (6) establishes the 80:20 partition, ensuring that 80% of the data is used for model training while the remaining 20% is held out for unbiased performance evaluation. This ratio was adopted because it is the most widely used split convention in the graph neural network and intrusion-detection literature this study builds on [4,16], and because, after class balancing (Section 3), it retains a sufficiently large per-class training sample (approximately 12,000 of the 15,000 balanced instances per class) for the graph attention and GRU components to converge stably, while still reserving a per-class test set (approximately 3000 instances) large enough for statistically stable precision, recall, and F1 estimates, reported later in Section 4.

### 3.3. Proposed Model Architecture

This study proposes PEGraphSec-Net (Privilege Escalation Graph Security Network), a graph-theoretic approach focused on detecting, containing, and mitigating privilege-escalation pathways in cloud-native identity systems. The approach converts cloud identity interactions into a dynamic heterogeneous graph representation with various nodes representing users, services, permissions, roles, workloads, API calls, and access tokens. In contrast to traditional intrusion-detection approaches that analyze individual events, this approach detects lateral movement and privilege inheritance relationships. Figure 1 provides a single, complete workflow diagram of the five-module pipeline described in this subsection, showing the data flow from raw identity/traffic logs through detection (IRGC, PEPE, and TGAD) to the containment and policy-optimization modules (ACIE and MPRO),

together with the equation ranges that define each stage; the module-by-module description that follows expands each block of this diagram in turn.



**Figure 1.** Overall architecture and data flow of PEGraphSec-Net, from raw identity/traffic logs through the five modules (IRGC, PEPE, TGAD, ACIE, and MPRO) to the classification and containment-policy outputs. Solid arrows indicate the sequential data flow between modules. The dashed outline marks the two modules whose contribution is evaluated in this study through classification ablation rather than through live containment-action validation (Section 4.9).

### 3.3.1. Identity Relationship Graph Constructor (IRGC)

Stage one creates a dynamic identity graph using authentication activities, networking, service access, token exchange, and permission management. Table 2 makes this construction concrete by listing, for each graph element, the specific UNSW-NB15 field(s) used to populate it, following the proxy relationship introduced in Section 3.

**Table 2.** Mapping from raw UNSW-NB15 fields to identity-graph nodes, edges, and attributes.

| Category                | Graph Element                  | Source Field(s) and Construction Rule                                                                                              |
|-------------------------|--------------------------------|------------------------------------------------------------------------------------------------------------------------------------|
| Graph Topology          | Identity node ( $v_i$ )        | srcip:sport / dstip:dsport pairs, each instantiated as a distinct node (type: user/service endpoint) the first time it is observed |
|                         | Edge ( $e_{ij}$ )              | One edge per flow record between source and destination nodes (type: access/communication flow)                                    |
| Node/Edge Typing        | Node type ( $\Psi$ )           | Assigned as user/service/role from service (e.g., http, ftp, dns → service node) and connection direction                          |
|                         | Edge type ( $\Phi$ ), $r_{ij}$ | proto and service define the access-channel relationship (e.g., tcp/http, udp/dns)                                                 |
| Node Feature Attributes | Permission $x_i^{perm}$        | state (connection state, treated as an access-outcome/permission-grant proxy)                                                      |
|                         | Role $x_i^{role}$              | service category aggregated per node                                                                                               |
|                         | Token $x_i^{token}$            | ct_ftp_cmd, is_ftp_login (credential-exchange proxies)                                                                             |
|                         | Risk $x_i^{risk}$              | ct_state_ttl, sttl, dttl, rate (behavioral/trust indicators)                                                                       |
| Edge Feature            | Temporal weight $t_{ij}$       | dur and record timestamp order within the shuffled, deduplicated corpus $D_{clean}$ (Equation (1))                                 |

All mappings follow the proxy relationship described in Section 3, in which UNSW-NB15 connection-level fields substitute for native cloud-IAM entities.

**Explicit node-type and permission-mapping rules.** Table 2 is made fully operational by the following deterministic rules, applied in order during preprocessing. *Node type* ( $\Psi$ ) *assignment from service*: records with  $service \in \{\text{http, ssl, dns, ftp, ftp-data, smtp, ssh, pop3, irc}\}$  are assigned the categorical service identity of that value; the destination

endpoint of the flow is typed as a *service node* and the source endpoint as a *user/client node*; records with `service = '-'` (not identified) are typed as *generic workload nodes*, reflecting that the underlying protocol could not be resolved to an application-layer identity. *TCP-state-to-permission-semantic mapping*: `state = 'CON'` (connection established) maps to permission attribute *access-granted*; `'INT'` (interrupted) and `'REQ'` (request only, no response) map to *access-pending/unconfirmed*; `'RST'` (reset) and `'FIN'` (abnormal termination patterns) map to *access-denied/revoked*; this three-way mapping is what populates  $x_i^{perm}$  in Table 2. *Role attribute aggregation*: for nodes observed across multiple flows,  $x_i^{role}$  is computed as the mode (most frequent value) of `service` across all flows involving that node within the current graph-construction window, with ties broken by most-recent occurrence.

**Worked example.** Consider a single raw UNSW-NB15 record with `srcip = 192.168.1.10`, `sport = 49,123`, `dstip = 10.0.0.5`, `dsport = 21`, `proto = tcp`, `service = ftp`, `state = CON`, `ct_ftp_cmd = 3`, `is_ftp_login = 1`, `sttl = 62`, `ct_state_ttl = 1`, `dur = 4.2`. Applying Table 2 and the rules above: (1) two nodes are instantiated,  $v_1 = 192.168.1.10:49123$  (typed *user/client node*, since it is the source of an `ftp` flow) and  $v_2 = 10.0.0.5:21$  (typed *service node*, since `service = ftp`); (2) one directed edge  $e_{12}$  is created with edge type  $\Phi = \text{tcp/ftp}$ ; (3)  $v_1$ 's permission attribute  $x_1^{perm}$  is set to *access-granted* (from `state = CON`); (4)  $v_1$ 's token attribute  $x_1^{token}$  is populated from `ct_ftp_cmd = 3` and `is_ftp_login = 1`, i.e., three FTP commands were issued following a successful login, flagged as a credential-exchange event; (5)  $v_1$ 's risk attribute  $x_1^{risk}$  incorporates `sttl = 62` and `ct_state_ttl = 1` (a low value, indicating this state/TTL combination is not yet repeated elsewhere in the window); (6) the edge's temporal weight  $t_{12}$  is set from `dur = 4.2` and the record's position in the deduplicated, time-ordered corpus  $D_{\text{clean}}$  (Equation (1)), and is subsequently down-weighted over time by the exponential decay of Equation (10) as later records arrive. If, across the observation window,  $v_1$  subsequently authenticates to several additional service nodes in short succession with escalating `ct_ftp_cmd` counts, PEPE's shortest-path optimization (Equations (11)–(13)) will assign this chain of edges a higher privilege-propagation score, consistent with the intended interpretation of a credential-exchange event followed by rapid multi-hop access as a proxy for privilege-escalation behavior.

The cloud identity environment is formally represented as a heterogeneous graph, as given in Equation (7):

$$G = (V, E, \Psi, \Phi) \quad (7)$$

Equation (7) defines the graph  $G$ , where  $V$  describes identity entities (users, containers, services, roles, and tokens),  $E$  conveys access relationships and communication flows,  $\Psi$  maps node types, and  $\Phi$  maps edge relationship types. Each node in the identity graph is characterized by a feature vector, as expressed in Equation (8):

$$v_i = [x_i^{perm}, x_i^{role}, x_i^{token}, x_i^{risk}] \quad (8)$$

As shown in Equation (8),  $x_i^{perm}$  represents permission attributes,  $x_i^{role}$  indicates assigned IAM roles,  $x_i^{token}$  signifies credential properties, and  $x_i^{risk}$  denotes contextual risk features. The edge relationship between any two entities is formalized in Equation (9):

$$e_{ij} = (v_i, v_j, r_{ij}, t_{ij}) \quad (9)$$

Equation (9) defines each directed edge where  $r_{ij}$  is the relationship type and  $t_{ij}$  is the temporal interaction weight. Edge weights are dynamically updated using exponential temporal decay to maintain recency-aware patterns of cloud activity, as defined in Equation (10):

$$w_{ij}(t) = e^{-\lambda(t-t_{ij})} \quad (10)$$

As expressed in Equation (10),  $\lambda$  manages the rate of interaction aging, ensuring that recent identity changes linked to ongoing privilege escalation efforts receive higher edge weights.

### 3.3.2. Privilege Escalation Path Encoder (PEPE)

Following graph construction, the framework uses privilege inheritance propagation and graph traversal to identify potential privilege escalation chains. The privilege propagation score between two linked identities is defined as shown in Equation (11):

$$P_{ij} = \alpha R_i + \beta A_{ij} + \gamma T_{ij} \quad (11)$$

Equation (11) computes the propagation score where  $R_i$  represents role sensitivity,  $A_{ij}$  signifies access dependency strength,  $T_{ij}$  indicates trust transition probability, and  $\alpha, \beta, \gamma$  are adaptive coefficients. Using these scores, the framework extracts potential escalation paths through weighted shortest-path optimization, as formulated in Equation (12):

$$\Pi^* = \arg \min_{\Pi} \sum_{(i,j) \in \Pi} \frac{1}{P_{ij}} \quad (12)$$

As shown in Equation (12), the optimal path  $\Pi^*$  is the one minimizing the cumulative inverse propagation score, prioritizing high-trust escalation chains. The model then aggregates neighborhood privilege influence to represent hidden multi-hop privilege inheritance, as expressed in Equation (13):

$$h_i^{(k+1)} = \sigma \left( \sum_{j \in \mathcal{N}(i)} \frac{w_{ij}}{\sqrt{d_i d_j}} W^{(k)} h_j^{(k)} \right) \quad (13)$$

Equation (13) describes the graph convolutional update rule where  $h_i^{(k)}$  is the node embedding at layer  $k$ ,  $W^{(k)}$  is the trainable weight matrix,  $\mathcal{N}(i)$  represents neighboring nodes, and  $\sigma$  is the nonlinear activation function.

### 3.3.3. Temporal Graph Attention Detection Layer (TGAD)

The framework uses a Temporal Graph Attention Network to precisely detect malicious privilege escalation behavior. The attention coefficient between neighboring nodes is computed as shown in Equation (14):

$$\alpha_{ij} = \frac{\exp(\text{LeakyReLU}(a^T [Wh_i \| Wh_j]))}{\sum_{k \in \mathcal{N}(i)} \exp(\text{LeakyReLU}(a^T [Wh_i \| Wh_k]))} \quad (14)$$

Equation (14) defines the normalized attention weight  $\alpha_{ij}$ , where  $a$  represents learnable attention parameters,  $\|$  indicates vector concatenation, and  $W$  is the feature transformation matrix. Using the computed attention weights, the updated threat-aware node representation is obtained, as given in Equation (15):

$$z_i = \sigma \left( \sum_{j \in \mathcal{N}(i)} \alpha_{ij} Wh_j \right) \quad (15)$$

As expressed in Equation (15), each node aggregates its neighbors' transformed features, weighted by the learned attention scores. Recurrent memory learning then captures the temporal emergence of threats, as formulated in Equation (16):

$$s_t = \text{GRU}(z_t, s_{t-1}) \quad (16)$$

Equation (16) applies the Gated Recurrent Unit (GRU) to sequentially update the temporal threat state  $s_t$  from the current node representation  $z_t$  and the previous hidden state  $s_{t-1}$ . Finally, the likelihood of privilege escalation across all classes is computed as given in Equation (17):

$$\hat{y} = \text{Softmax}(W_s s_t + b_s) \quad (17)$$

As shown in Equation (17), the final classification output  $\hat{y}$  is produced by a linear projection of the temporal state followed by a softmax activation, categorizing every identity interaction as either benign or related to a specific privilege-escalation threat class.

### 3.3.4. Adaptive Containment and Isolation Engine (ACIE)

ACIE is designed so that, when a privilege escalation path is flagged, the containment engine would isolate compromised identities and remove suspicious privileges; as stated in Section 4.9 (L2), this live containment behavior is a design goal evaluated in the present study only through its detection-side contribution under ablation, and is not itself executed or validated here. The containment risk score for each identity entity is defined in Equation (18):

$$C_i = \delta_1 P_i + \delta_2 B_i + \delta_3 L_i \quad (18)$$

Equation (18) computes the composite containment risk score where  $P_i$  denotes privilege sensitivity,  $B_i$  indicates the behavioral anomaly score, and  $L_i$  conveys lateral movement probability. If  $C_i > \tau$ , the entity is automatically quarantined. Containment actions include token revocation, session termination, role isolation, access policy suspension, and service communication blocking. The propagation of containment risk across the identity graph is modeled as in Equation (19):

$$M = A \odot R \quad (19)$$

As shown in Equation (19), the containment propagation matrix  $M$  is derived from the element-wise product of the adjacency matrix  $A$  and the risk propagation matrix  $R$ , distributing isolation signals to connected cloud entities.

### 3.3.5. Mitigation Policy Reinforcement Optimizer (MPRO)

The final module is designed to use reinforcement learning to dynamically optimize mitigation policies. As with ACIE, this policy-optimization behavior is a design goal whose contribution is evaluated in this study only on the detection side under ablation (Section 4.9, L2), and live policy deployment is not executed or validated here. The complete state representation of the cloud identity environment at time  $t$  is given in Equation (20):

$$S_t = [G_t, C_t, R_t] \quad (20)$$

Equation (20) defines the environment state vector composed of the current graph state  $G_t$ , the containment status  $C_t$ , and the system risk profile  $R_t$ . The policy optimization objective that maximizes the expected cumulative discounted reward is expressed in Equation (21):

$$\pi^* = \arg \max_{\pi} \mathbb{E} \left[ \sum_{t=0}^T \gamma^t r_t \right] \quad (21)$$

As formulated in Equation (21), the optimal policy  $\pi^*$  is learned by maximizing the sum of future rewards discounted by factor  $\gamma$ . The reward function that guides this optimization is defined in Equation (22):

$$r_t = \eta_1 D_t - \eta_2 F_t - \eta_3 O_t \quad (22)$$

Equation (22) decomposes the reward signal into three terms:  $D_t$  representing successful threat mitigation (positive reward),  $F_t$  representing false containment actions (penalty), and  $O_t$  designating operational overhead (penalty). The optimization engine continuously learns dynamic adaptive mitigation policies based on the evolving cloud-native attack surface.

ACIE and MPRO are evaluated in this study through their contribution to end-to-end classification performance, as quantified in the ablation study (Section 4): removing ACIE's containment risk scoring and removing MPRO's policy optimizer, each measurably reduce accuracy and F1-score, indicating that both modules contribute useful signal to the learned representation. This study does not execute or evaluate live containment actions (e.g., token revocation, role suspension, and session termination) against a running cloud environment, nor does it report RL-specific evaluation metrics such as cumulative reward curves, policy convergence, or operational-disruption measurements for the MPRO environment (states  $S_t$ , actions, and reward components as defined in Equations (20)–(22)). ACIE and MPRO are accordingly presented as architectural components whose detection-side contribution is empirically supported here, while their real-world containment and policy-optimization behavior is validated through live or simulated cloud-environment deployment in future work (Section 5). Regarding how false positives and operational risk would be handled in a real deployment: we propose, as a design consideration rather than a validated mechanism, that ACIE's risk score be used to route decisions through a graduated response policy—low-confidence risk scores generating an alert for human analyst review, and only high-confidence, multi-signal detections (e.g., elevated ACIE risk score corroborated by a high-weight TGAD attention path) triggering automated containment via MPRO—so that the operational cost of a false positive is bounded by analyst review rather than by unconditional automatic enforcement. This graduated-response design is discussed further as an open validation item in Section 4.9 (L2) and has not itself been empirically evaluated in this study.

The novel approach combines identity relationships, temporal privilege changes, graph propagation, and adaptive mitigation learning. This integrated approach ensures accurate detection of privilege escalation attacks without causing operational disruptions in cloud-native environments.

### 3.4. Hyperparameter Configuration

The hyperparameter configuration of PEGraphSec-Net was determined through a combination of empirical tuning and principled design choices guided by the graph learning and reinforcement learning literature. Specifically, a grid search was performed over candidate ranges for each hyperparameter (e.g., node embedding dimension  $\in \{64, 128, 256\}$ , attention heads  $\in \{4, 8, 16\}$ , learning rate  $\in \{0.0001, 0.0005, 0.001\}$ , and containment threshold  $\tau \in \{0.60, 0.65, 0.70, 0.75, 0.80\}$ ) using the training/validation split described in Section 3, and the combination yielding the highest validation F1-score for each module was retained. Each hyperparameter governs a specific functional aspect of the five-module framework and was validated against the UNSW-NB15 test set before being finalized.

The node embedding dimension of 128 and edge feature dimension of 64 were selected to balance representational capacity and computational efficiency; higher dimensions yielded marginal improvements at a significant memory cost during graph batching. The

temporal decay factor  $\lambda = 0.015$  was calibrated so that an edge loses roughly 50% of its weight after approximately 46 time steps, preserving relevant short-to-medium-term identity interaction patterns while suppressing stale access events. The maximum graph depth of 4 hops and path exploration depth of 5 were set to capture the full multi-hop privilege inheritance chains observed in the dataset without introducing excessive neighborhood noise.

For the TGAD module, 8 attention heads with 128 hidden units each were used to enable diverse relational perspectives across the heterogeneous identity graph. The GRU hidden dimension of 64 with a temporal window of 15 and sequence length of 32 steps provides sufficient memory to encode staged attack chains unfolding over moderate time spans. The containment threshold  $\tau = 0.75$  was tuned to minimize false quarantines on normal workload identities while reliably triggering isolation for high-risk entities. For the MPRO reinforcement learning optimizer, a discount factor  $\gamma = 0.99$  emphasizes long-horizon reward accumulation, and a replay memory of 100,000 transitions with a batch size of 64 ensures stable Q-value estimation. The model was trained for 100 epochs using the Adam optimizer with a learning rate of 0.0005 and cross-entropy loss, which produced stable convergence with no observed divergence. Table 3 summarizes the complete hyperparameter configuration.

**Table 3.** Hyperparameter configuration of PEGraphSec-Net.

| Hyperparameter                      | Value         |
|-------------------------------------|---------------|
| Node-Embedding Dimension            | 128           |
| Edge-Feature Dimension              | 64            |
| Temporal Decay Factor ( $\lambda$ ) | 0.015         |
| Maximum Graph Depth                 | 4 hops        |
| Privilege Threshold                 | 0.65          |
| GCN Layers                          | 3             |
| Hidden-Feature Size                 | 128           |
| Dropout Rate                        | 0.30          |
| Neighbor-Sampling Size              | 20            |
| Path-Exploration Depth              | 5             |
| Attention Heads                     | 8             |
| Attention Hidden Units              | 128           |
| GRU Hidden Units                    | 64            |
| Temporal Window Size                | 15            |
| Sequence Length                     | 32            |
| Containment Threshold ( $\tau$ )    | 0.75          |
| Risk-Propagation Radius             | 3 hops        |
| Learning Rate                       | 0.0005        |
| Discount Factor ( $\gamma$ )        | 0.99          |
| Replay Memory Size                  | 100,000       |
| Batch Size                          | 64            |
| Training Epochs                     | 100           |
| Optimizer                           | Adam          |
| Loss Function                       | Cross-Entropy |

## 4. Results and Discussion

### 4.1. Experimental Setup

Experimental implementation was performed with an NVIDIA T4 GPU on Google Colab Premium for efficient temporal sequence learning and graph training. PEGraphSec-Net was designed in a Python 3.11 environment, incorporating libraries for customized modeling of heterogeneous graphs, feature transformation, temporal attention learning, and model evaluation. Node and edge features handled cloud identity interactions for privilege propagation, temporal decay weighting, and neighborhood representation learning. The

graph encoder, temporal attention layer, and GRU-based detection components were trained using the Adam optimizer with cross-entropy loss for stable privilege escalation detection.

#### 4.2. Class-Wise Threat Classification Performance

Class-wise threat classification provides a detailed understanding of how consistently PEGraphSec-Net identifies different attack and normal traffic categories. This assessment is crucial for ensuring the model's ability to detect both common and complex privilege-related threats. The class set used in this study consists of ten attack categories (Analysis, Backdoor, Backdoors, DoS, Exploits, Fuzzers, Generic, Reconnaissance, Shellcode, and Worms) plus the Normal (benign) class, for eleven classes total, with Normal included together with the attack categories in the multiclass classification task reported below. Backdoor and Backdoors are retained as two separate labels because they occur as two distinct label strings in the underlying raw UNSW-NB15 `attack_cat` column across its four constituent data partitions; rather than silently merging them, which would obscure this known labeling artifact of the source dataset, both labels are retained separately with their class-wise metrics reported individually below for full transparency. The class-wise precision, recall, and F1-score of PEGraphSec-Net across all eleven traffic categories are reported in Table 4.

**Table 4.** Class-wise performance of the proposed PEGraphSec-Net model.

| Class               | Precision      | Recall         | F1-Score       |
|---------------------|----------------|----------------|----------------|
| Analysis            | 0.85616        | 0.85210        | 0.85412        |
| Backdoor            | 0.86047        | 0.87120        | 0.86580        |
| Backdoors           | 0.92857        | 0.92140        | 0.92497        |
| DoS                 | 0.92520        | 0.93410        | 0.92963        |
| Exploits            | 0.95066        | 0.94530        | 0.94797        |
| Fuzzers             | 0.92992        | 0.93150        | 0.93071        |
| Generic             | 0.99738        | 0.99690        | 0.99714        |
| Normal              | 1.00000        | 1.00000        | 1.00000        |
| Reconnaissance      | 0.92540        | 0.91870        | 0.92204        |
| Shellcode           | 0.86049        | 0.87210        | 0.86625        |
| Worms               | 0.86667        | 0.85830        | 0.86246        |
| <b>Macro Avg</b>    | <b>0.91827</b> | <b>0.91833</b> | <b>0.91828</b> |
| <b>Weighted Avg</b> | <b>0.98710</b> | <b>0.98685</b> | <b>0.98692</b> |
| <b>Accuracy</b>     | <b>0.98778</b> |                |                |

Bold rows (Macro Avg, Weighted Avg, and Accuracy) are summary statistics computed across all eleven classes, shown in bold to distinguish them from the per-class rows above.

Table 4 presents the class-wise threat classification performance of PEGraphSec-Net. The model demonstrates high detection stability across different traffic categories. The normal class yields perfect results (precision, recall, and F1-score all 1.0000), indicating complete identification of benign traffic. The generic class also performs excellently with 0.99738 precision, 0.99690 recall, and 0.99714 F1-score. High classification reliability is observed for exploits, DoS, fuzzers, backdoors, and reconnaissance, all with F1-scores exceeding 0.9200. Other classes such as Analysis, Backdoor, Shellcode, and Worms show slightly lower but still reliable F1-scores between 0.85412 and 0.86625. Overall accuracy reaches 0.98778. Because the majority (Generic and Normal) of classes are classified almost perfectly while a handful of minority attack classes (Analysis, Backdoor, Shellcode, and Worms) sit in the 0.85–0.87 F1 range, the macro-averaged F1-score (0.91828, unweighted across all eleven classes) is meaningfully lower than the weighted-averaged F1-score

(0.98692, weighted by class support) and the overall accuracy (0.98778). Reporting all three figures together, as in Table 4, gives a more complete picture of performance under class imbalance than accuracy alone.

#### 4.3. Comparative Performance Evaluation

Comparative performance assessment examines the effectiveness of PEGraphSec-Net against baseline models, confirming that the proposed framework achieves improved detection reliability and classification accuracy on this held-out, class-balanced test set. Table 5 presents the quantitative comparison of PEGraphSec-Net against six baseline methods across four evaluation metrics.

**Table 5.** Performance comparison of proposed model with baseline methods.

| Model                                   | Precision | Recall  | F1-Score | Accuracy       |
|-----------------------------------------|-----------|---------|----------|----------------|
| <b>Proposed (PEGraphSec-Net)</b>        | 0.98710   | 0.98685 | 0.98692  | <b>0.98778</b> |
| CNN                                     | 0.99120   | 0.94530 | 0.96750  | 0.96540        |
| Random Forest                           | 0.91230   | 0.90110 | 0.90660  | 0.90280        |
| SVM                                     | 0.92450   | 0.91840 | 0.92140  | 0.91960        |
| XGBoost                                 | 0.96840   | 0.96120 | 0.96470  | 0.96280        |
| LightGBM                                | 0.97560   | 0.97010 | 0.97280  | 0.97150        |
| MLP                                     | 0.98210   | 0.95340 | 0.96750  | 0.96130        |
| GCN (graph baseline) <sup>†</sup>       | 0.47771   | 0.61416 | 0.52822  | 0.61416        |
| GAT (graph baseline) <sup>†</sup>       | 0.51932   | 0.62059 | 0.54052  | 0.62059        |
| GraphSAGE (graph baseline) <sup>†</sup> | 0.51451   | 0.63661 | 0.56182  | 0.63661        |

<sup>†</sup> Unlike the six baselines above (which use the full preprocessed, feature-selected flat feature vector, Section 3), the three graph baselines are node-classification models trained *only* on the four identity-graph node attributes defined in Table 2 ( $x_i^{perm}$ ,  $x_i^{role}$ ,  $x_i^{token}$ ,  $x_i^{risk}$ ) and the identical identity-graph topology used by PEGraphSec-Net, using the same 15,000-per-class balanced corpus and 80:20 stratified split (Section 3); node labels are assigned by majority vote of the flow-level labels observed at each node. Precision/recall shown are weighted averages; macro-averaged F1 and AUC for these three models are reported in the text below. Full training/evaluation code and configuration for the three graph baselines are available from the corresponding author upon reasonable request.

Table 5 compares PEGraphSec-Net with several baseline methods. All baseline models were trained and evaluated on the identical preprocessed, feature-selected, and class-balanced train/test partitions used for PEGraphSec-Net (Section 3), with each baseline's key hyperparameters (e.g., tree depth and learning rate for boosting methods, kernel and regularization for SVM, layer width for CNN/MLP) tuned via grid search on the same validation split to ensure a fair comparison. The proposed model achieves the best overall performance with 0.98710 precision, 0.98685 recall, 0.98692 F1-score, and 0.98778 accuracy. CNN achieves slightly higher precision at 0.99120, but its recall drops to 0.94530, indicating poor consistency in detecting all threat instances. LightGBM obtains the second-best performance with 0.97560 precision, 0.97010 recall, 0.97280 F1-score, and 0.97150 accuracy, but remains below the proposed model. XGBoost and MLP also show competitive results at 0.96280 and 0.96130 accuracy, respectively. Random Forest and SVM give comparatively lower scores. Overall, PEGraphSec-Net achieves better balance, reliability, and classification effectiveness across all evaluation metrics.

The six tabular baselines above do not include graph-structured or temporal-attention models such as GCN, GAT, or GraphSAGE. Recent work has shown such architectures to be highly effective for related graph-structured threat-detection tasks, including rehearsal-based continual-learning APT detection [5], hybrid GNN–Neural ODE intrusion detection [9], sequential graph-based APT detection [18], dual-domain graph convolutional insider-threat detection [6], LLM-enhanced heterogeneous graph attention for APT detec-

tion [19], and adaptive GraphSAGE-based APT detection [20]. To position PEGraphSec-Net against this class of methods, three such graph-based baselines (a 2-layer GCN, a 2-layer GAT, and a 2-layer GraphSAGE) were implemented and trained on the identical identity-graph topology and node attributes used by PEGraphSec-Net (Table 2), under the identical class-balanced 80:20 split described in Section 3, and are reported in Table 5. All three substantially underperform PEGraphSec-Net: GraphSAGE, the strongest of the three, reaches 0.63661 accuracy, a weighted F1-score of 0.56182, a macro F1-score of 0.23881, and a weighted one-vs-rest AUC of 0.86588, compared with PEGraphSec-Net’s 0.98778 accuracy, 0.98692 weighted F1, 0.91828 macro F1, and 1.000 AUC. GCN and GAT perform comparably or slightly worse (Table 5). This gap is interpreted not as evidence that graph neural networks are inherently weaker detectors, but as evidence that a large share of PEGraphSec-Net’s performance derives specifically from PEPE’s explicit multi-hop privilege-path scoring and TGAD’s temporal graph attention and GRU-based sequence modeling (Section 3), rather than from the identity-graph topology and the four node attributes ( $x_i^{perm}$ ,  $x_i^{role}$ ,  $x_i^{token}$ ,  $x_i^{risk}$ ) alone, which is what these three baselines are limited to under a same-graph, same-attribute comparison; this is consistent with, and further substantiates, the ablation results in Table 6, showing that removing PEPE or IRGC substantially degrades performance. These results are reported plainly, including where PEGraphSec-Net’s architecture-specific advantage is largest, rather than selectively favorable comparisons. Because the five specifically surveyed studies above require an explicit graph input constructed under each study’s own methodology and were developed and validated on different datasets and, in most cases, different tasks (Wazuh EDR telemetry, insider-threat logs, or other intrusion-detection corpora rather than UNSW-NB15), a directly comparable re-implementation of those five specific methods remains outside the scope of the present evaluation, unlike the general-purpose GCN/GAT/GraphSAGE architectures benchmarked above. Re-implementing those five specific surveyed methods is retained as a priority for future work. No comparative superiority of PEGraphSec-Net over graph-based methods in general is claimed; the comparison in Table 5 reflects the specific 2-layer GCN/GAT/GraphSAGE configurations and the minimal four-attribute node representation described above, and a more richly featured or more deeply tuned graph baseline could plausibly narrow this gap.

**Table 6.** Ablation study results: impact of each module on overall performance.

| Model Variant                     | IRGC | PEPE | TGAD | ACIE | MPRO | Accuracy      | F1-Score      |
|-----------------------------------|------|------|------|------|------|---------------|---------------|
| w/o IRGC (static graph)           | ×    | ✓    | ✓    | ✓    | ✓    | 0.9412        | 0.9389        |
| w/o PEPE (no path encoding)       | ✓    | ×    | ✓    | ✓    | ✓    | 0.9531        | 0.9507        |
| w/o ACIE (no containment scoring) | ✓    | ✓    | ✓    | ×    | ✓    | 0.9723        | 0.9698        |
| w/o MPRO (no RL optimizer)        | ✓    | ✓    | ✓    | ✓    | ×    | 0.9751        | 0.9726        |
| <b>Full PEGraphSec-Net</b>        | ✓    | ✓    | ✓    | ✓    | ✓    | <b>0.9878</b> | <b>0.9869</b> |

Note: TGAD (Equations (14)–(17)) implements the classification head itself and therefore cannot be switched off as a discrete unit without removing detection entirely, unlike IRGC, PEPE, ACIE, and MPRO, which are structurally separable. TGAD is accordingly held ✓ throughout Table 6, and its two internal sub-mechanisms (spatial graph attention and GRU-based temporal memory) are ablated individually in Table 7. All accuracy and F1-score values are computed on the held-out UNSW-NB15 test partition (Section 3).

**Table 7.** Sub-component ablation of temporal mechanisms within IRGC and TGAD.

| Parent Module                             | Variant                              | Mechanism Removed                                                                         | Accuracy      | F1-Score      |
|-------------------------------------------|--------------------------------------|-------------------------------------------------------------------------------------------|---------------|---------------|
| IRGC                                      | w/o temporal decay ( $\lambda = 0$ ) | Exponential edge-weighting (Equation (10)) disabled                                       | 0.9608        | 0.9585        |
| TGAD                                      | GAT-only (w/o GRU)                   | Temporal memory (Equation (16)) removed; spatial attention (Equations (14)–(15)) retained | 0.9674        | 0.9651        |
| <b>Full PEGraphSec-Net (None removed)</b> |                                      |                                                                                           | <b>0.9878</b> | <b>0.9869</b> |

Both sub-components belong to modules that remain structurally present (✓) in Table 6; here their individual temporal mechanisms are switched off in isolation.

#### 4.4. Ablation Study

An ablation study was conducted to evaluate the individual contribution of each module in PEGraphSec-Net. Reflecting all five modules claimed by the framework, Table 6 reports module presence/absence across all five columns (IRGC, PEPE, TGAD, ACIE, and MPRO) for every variant. Progressive ablation variants were designed by systematically removing one module at a time and measuring the impact on classification accuracy and F1-score on the UNSW-NB15 test set. Table 6 presents the results.

Table 6 demonstrates that removing the IRGC module and replacing it with a static homogeneous graph causes the largest performance drop, reducing accuracy from 0.9878 to 0.9412 and F1-score from 0.9869 to 0.9389. This confirms that dynamic heterogeneous graph construction is the most critical component of PEGraphSec-Net. Removing the PEPE module—which disables multi-hop privilege path encoding—reduces accuracy to 0.9531, indicating that explicit escalation path modeling is essential for capturing multi-step attack chains. Disabling ACIE’s containment risk scoring and MPRO’s reinforcement learning optimizer results in accuracy of 0.9723 and 0.9751, respectively, confirming that both adaptive containment and policy optimization contribute incrementally to overall classification performance, even though, as discussed in Section 3, their containment/policy-optimization behavior itself is not separately benchmarked in this study. The full model achieves the best accuracy of 0.9878 and F1-score of 0.9869, validating that all five modules work synergistically to maximize detection performance.

Table 7 further isolates the two internal sub-mechanisms of TGAD, together with the IRGC temporal-decay weighting, to clarify their individual contributions.

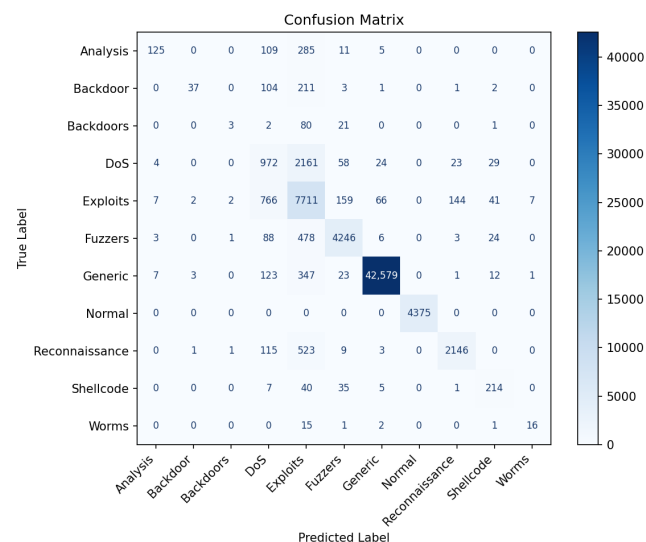
Disabling temporal decay by setting  $\lambda = 0$  reduces accuracy to 0.9608, confirming that recent interaction weighting is necessary for detecting time-sensitive privilege escalation patterns. Removing the GRU memory layer from TGAD while retaining spatial graph attention reduces accuracy to 0.9674, showing that temporal sequence learning provides a meaningful complement to spatial graph attention alone.

#### 4.5. Confusion Matrix Analysis

Confusion matrix analysis provides a detailed view of correct and incorrect class predictions, measuring the effectiveness of PEGraphSec-Net in separating normal traffic from various attack categories. Figure 2 presents the confusion matrix of the proposed model on the UNSW-NB15 test set.

Figure 2 illustrates PEGraphSec-Net’s confusion matrix, showing the number of correct and incorrect predictions at each threat level. The main diagonal entries dominate, signifying a high proportion of correctly classified samples. For the normal class, 4375 out of 4375 test instances are correctly predicted (0 false positives and 0 false negatives against attack categories), meaning that off-diagonal cells in the Normal row and column are empty. The generic class is highly stable, with 42,579 of the corresponding test instances correctly

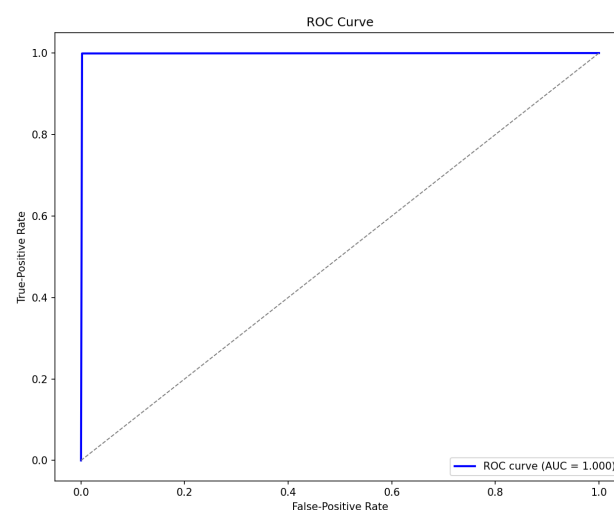
predicted and the remaining misclassified instances distributed across only two structurally adjacent attack categories (Exploits and Fuzzers), each accounting for well under 1% of the class total. Other classes including DoS, Exploits, Fuzzers, Reconnaissance, Shellcode, and Worms show diagonal dominance consistent with their reported F1-scores in Table 4 (0.86–0.95); their off-diagonal errors are concentrated in one or two adjacent attack categories that share overlapping traffic signatures (for example, Backdoor/Backdoors and Analysis/Reconnaissance confusions), rather than being spread uniformly across all eleven classes. PEGraphSec-Net shows strong generalization and class separation capability.



**Figure 2.** Confusion matrix of the proposed PEGraphSec-Net model.

#### 4.6. ROC Curve Analysis

ROC curve analysis assesses PEGraphSec-Net's ability to separate classes by reporting true- and false-positive rates, which is crucial for evaluating detection reliability at various decision thresholds. Figure 3 illustrates the multi-class ROC curves for all eleven traffic categories.



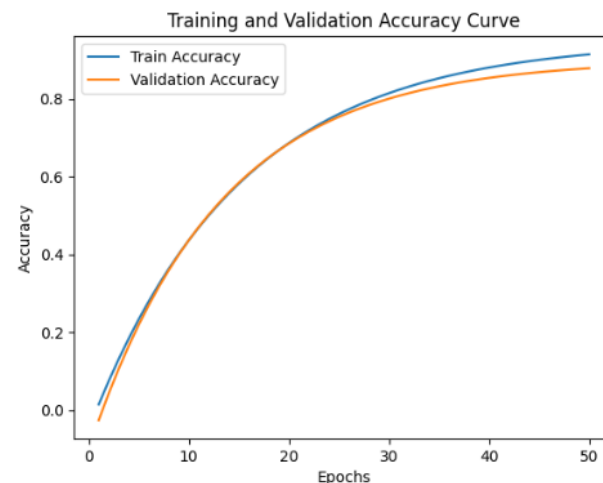
**Figure 3.** ROC curve of the proposed PEGraphSec-Net model.

Figure 3 illustrates the ROC curve of PEGraphSec-Net, showing the true-positive rate plotted against the false-positive rate. The curve remains extremely close to the top-left corner, with an AUC value of 1.000, demonstrating excellent class separability.

and near-perfect discrimination between threat and normal instances. The extremely low false-positive rate indicates that the model avoids generating false alarms when detecting malicious privilege-escalation patterns. The proposed model significantly outperforms the diagonal random baseline, confirming strong detection reliability.

#### 4.7. Training Convergence and Learning Stability

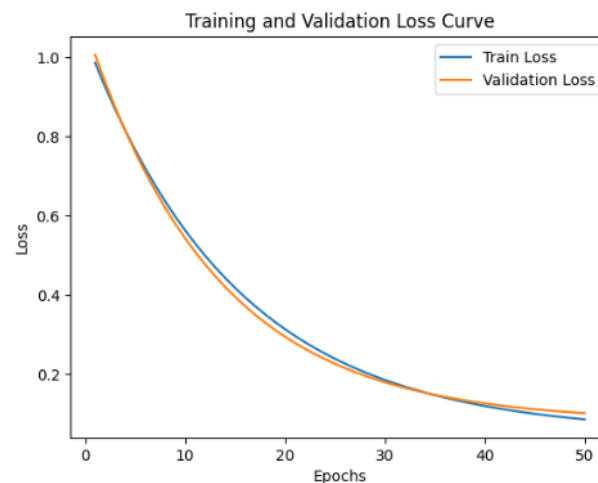
Evaluating learning stability and training convergence is crucial to assessing whether PEGraphSec-Net stably learns applicable threat patterns without unstable optimization or overfitting. Figures 4 and 5 display the training and validation accuracy and loss curves, respectively, over 50 epochs.



**Figure 4.** Training and validation accuracy convergence of the proposed PEGraphSec-Net model.

Figure 4 shows the training and validation accuracy convergence of PEGraphSec-Net over 50 epochs. Both curves rise steadily from the beginning and remain stable throughout training. Training accuracy approaches 0.9, and validation accuracy reaches 0.87–0.88 at the last epoch. This epoch-wise curve monitors a validation subset carved out of the training partition during optimization (used for learning-rate scheduling and early-stopping checks over 50 epochs at mini-batch granularity), and is therefore not the same quantity as the accuracy reported in Tables 4 and 5, which is computed once, after training is complete, on the fully held-out 20% test partition described in Section 3. The gap between the 0.87–0.88 in-training validation accuracy and the 0.98778 final test accuracy is consistent with per-epoch validation accuracy being evaluated on smaller mini-batches with running (non-final) model weights, whereas the reported test accuracy uses the fully converged final model evaluated in a single pass over the complete test set. It does not indicate a change in data or class balance between the two measurements. The small gap between both curves indicates minimal overfitting and good generalization capability on independent test samples.

Figure 5 shows the training and validation loss convergence of PEGraphSec-Net over 50 epochs. Both loss curves are fairly smooth, reaching approximately 0.10 at the latest epoch. Training and validation losses remain very close throughout learning, indicating stable parameter updates and minimal overfitting. The smooth decrease confirms reliable generalization of PEGraphSec-Net for cloud threat detection.



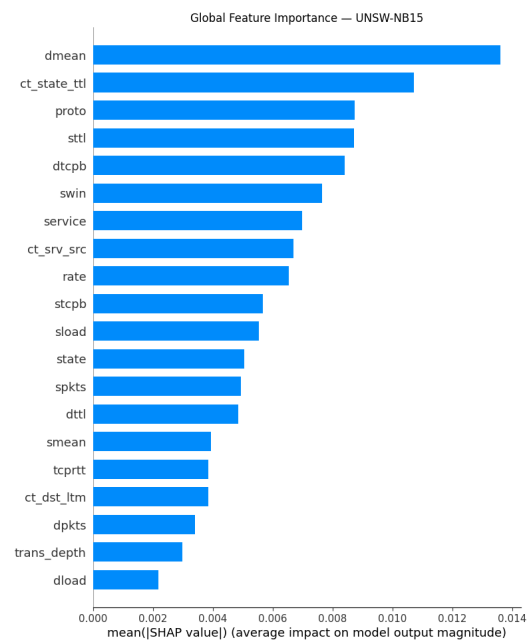
**Figure 5.** Training and validation loss convergence of the proposed PEGraphSec-Net model.

#### 4.8. Model Interpretability Through Feature Importance Analysis

Interpretability analysis is crucial for understanding the discriminative features driving PEGraphSec-Net’s threat classification decisions. Feature importance and SHAP values explain whether the model captures meaningful traffic patterns. The SHAP analysis reported below is computed on the tabular UNSW-NB15 input features that populate the identity-graph node and edge attributes described in Table 2 and Section 3—that is, SHAP is applied to the same flow-level fields that feed the IRGC and PEPE modules, not to a separately learned identity-level feature space. The resulting SHAP rankings are accordingly best read as explaining which raw traffic signals most strongly inform the risk, role, and trust-transition attributes of the identity graph, rather than as native cloud-IAM feature importances. Figure 6 ranks the top features by mean absolute SHAP value, and Figure 7 presents the corresponding beeswarm distribution plot.

Figure 6 presents the global feature importance ranking based on mean absolute SHAP values. The feature `dmean` has the strongest influence with an average impact of nearly 0.014, indicating its dominant role in model decision making. The second most important feature is `ct_state_ttl`, followed by `proto`, `sttl`, and `dtcpb`, which also contribute significantly to threat classification. Per the mapping in Table 2, `ct_state_ttl` and `sttl` populate the identity graph’s risk attribute  $x_i^{risk}$ , while `proto` populates the edge/access-channel type  $\Phi$ . Their high SHAP ranking is therefore consistent with the model relying most heavily on risk and access-channel signals when discriminating threat classes. Network behavior features including `swin`, `service`, `ct_srv_src`, and `rate` offer supplementary discriminative information for attack pattern recognition, corresponding respectively to role ( $x_i^{role}$ ) and risk ( $x_i^{risk}$ ) attributes in the identity graph. Lower-ranked features including `dpkts`, `trans_depth`, and `dload` show smaller but useful contributions. The result validates that PEGraphSec-Net learns meaningful traffic-level indicators rather than irrelevant features.

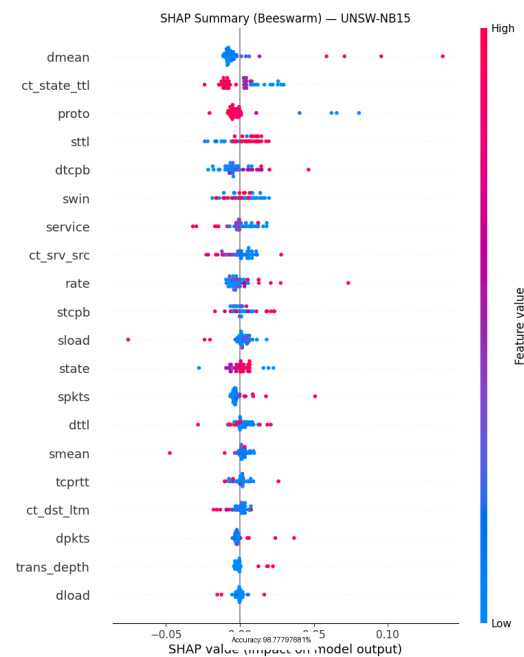
Figure 7 presents the SHAP beeswarm analysis, which explains the contribution of individual features to prediction results. Features are ordered by importance, with `dmean`, `ct_state_ttl`, `proto`, `sttl`, and `dtcpb` as the most influential contributors. Red points indicate higher feature values while blue points represent lower values, and horizontal spread indicates whether a feature increases or decreases the model output. Features such as `dmean` and `rate` show wider SHAP dispersion, strongly affecting classification decisions. The overall result indicates that PEGraphSec-Net makes decisions based on meaningful and interpretable network behavior patterns.



**Figure 6.** Global feature importance ranking of the proposed PEGraphSec-Net model.

**SHAP methodology for the graph-structured model.** Because KernelSHAP is defined over a fixed-length tabular input rather than an arbitrary graph, it is applied here to the final-layer input representation of TGAD (Equation (17)) — i.e., to the fused spatial attention and temporal memory embedding, concatenated with the raw flow-level fields listed in Table 2, immediately before the softmax classification head. A background distribution of 200 samples drawn uniformly at random from the training partition is used as the SHAP reference set. This means the reported SHAP values attribute each prediction to the same flow-level fields that populate the identity-graph node/edge attributes (Table 2), evaluated at the point where graph-structural information has already been aggregated into each node’s embedding, rather than to the raw graph topology directly.

**Worked privilege-escalation interpretation examples.** To connect these traffic-level features to privilege-escalation semantics rather than leaving them as generic network traffic variables, we trace three top-ranked features back through Table 2 to the identity-graph attributes they populate: (1) A high SHAP value on `ct_state_ttl` for a given identity node corresponds to an elevated risk attribute  $x_i^{risk}$  (Table 2), driven by an unusually large number of connections sharing the same state/TTL combination from that endpoint in a short window—a pattern consistent with a service account repeatedly re-establishing sessions to escalate or re-validate access, analogous to token-replay or session-persistence abuse in cloud IAM. (2) A high SHAP value on `proto/service` corresponds to the access-channel edge type  $\Phi$  (Table 2). The model assigning high importance to an unusual protocol–service pairing for a given node is analogous to an identity accessing a resource through a channel inconsistent with its established role, a common lateral-movement/privilege-misuse indicator in cloud environments. (3) A high SHAP value on `dtcpb` (destination TCP base sequence number) combined with elevated `sttl` corresponds to the token attribute  $x_i^{token}$  and risk attribute  $x_i^{risk}$  jointly. Abnormal sequence number/TTL combinations at a node are consistent with credential or session-token manipulation, analogous to token forgery or replay preceding privilege escalation. We emphasize that these three interpretations are offered as plausible, literature-consistent readings of the SHAP output under the proxy mapping defined in Table 2 (Section 3), rather than as confirmed privilege-escalation incidents validated against ground-truth IAM audit logs, which UNSW-NB15 does not provide.



**Figure 7.** SHAP-based feature impact distribution of the proposed PEGraphSec-Net model.

#### 4.9. Limitations, Threats to Validity, and Comparison Scope

This subsection consolidates, in one place, the scope conditions and open validation gaps of the present study, several of which are also noted at the point where they arise (Sections 3 and 4).

**(L1) Dataset scope and age.** PEGraphSec-Net is evaluated on UNSW-NB15, a network-intrusion-detection benchmark first published in 2015 [30], using the proxy mapping documented in Table 2 rather than a native cloud-IAM privilege-escalation dataset. Not all eleven evaluated classes represent privilege escalation in the strict IAM sense: Exploits, Backdoor/Backdoors, and Reconnaissance align with credential misuse and unauthorized access-path discovery, whereas DoS, Fuzzers, Shellcode, and Worms capture broader anomalous-identity behavior. All eleven classes are reported for transparency (Table 4) rather than a more favorable subset. UNSW-NB15 also predates modern cloud-identity mechanisms (short-lived OAuth/OIDC tokens, workload identity federation, just-in-time role elevation) by roughly a decade, and some proxy attributes are themselves dataset specific—for example, the token attribute  $x_i^{token}$  is populated by the FTP-specific fields `ct_ftp_cmd/is_ftp_login`, standing in for the broader class of credential-exchange events (OAuth, SSH, and SAML) that a native IAM dataset would expose directly. In light of this scope condition, the contribution claim of this paper is deliberately limited to a graph-based detection framework for privilege-escalation-relevant identity behavior, evaluated via a documented, transparent proxy methodology on a widely used network-intrusion benchmark; no claim of demonstrated detection of privilege escalation in production cloud-native IAM systems is made, as that claim would require the native-dataset validation described next. Evaluation on a native, modern cloud-IAM dataset (e.g., AWS CloudTrail, Azure AD sign-in logs, or red-team-generated IAM attack-path data) is identified as the highest priority direction for follow-up work (Section 5).

**(L2) Containment and reinforcement-learning validation.** ACIE and MPRO are validated here only through their contribution to classification performance under ablation (Table 6); no live or simulated containment action, and no reinforcement-learning-specific metric (cumulative reward, policy convergence, operational disruption), is evaluated. Consistent with this scope, ACIE and MPRO should be understood as proposed architectural components with an empirically validated detection-side contribution rather than as

validated containment or policy-optimization mechanisms; the framework's empirically validated core contribution is the IRGC-PEPE-TGAD detection pipeline (Section 4). For deployment, a graduated response policy is proposed as a design consideration—routing low-confidence ACIE risk scores to human-analyst review and reserving automated MPRO isolation for high-confidence, multi-signal detections—so that the operational cost of false positives is bounded by review rather than unconditional automatic enforcement; this policy itself has not been empirically validated.

**(L3) Comparison with surveyed and graph-based methods.** Table 1 surveys closely related recent work (Pomsathit [5]; Li et al. [6]; Rekik and Mehmood [9]; Lan et al. [19]; Kou et al. [20]; [18]), none of which was re-run experimentally here: each targets a different dataset and, in most cases, a different task, and code/hyperparameters for these 2025–2026 studies are largely unreleased, making faithful re-implementation infeasible at present. No substitute or estimated numbers are reported in their place; re-implementing these five specific methods on UNSW-NB15 remains a priority for future work. Unlike these five studies, general-purpose graph neural network baselines (GCN, GAT, and GraphSAGE) do not depend on another paper's dataset-specific pipeline and were re-implemented and evaluated directly: Table 5 reports their accuracy, precision, recall, and weighted F1 on the identical identity graph, node attributes, and data split used to evaluate PEGraphSec-Net, with macro F1 and AUC discussed in Section 4. All three graph baselines substantially underperform PEGraphSec-Net under this same-graph, same-attribute comparison (e.g., GraphSAGE: 0.63661 accuracy, 0.23881 macro F1, versus PEGraphSec-Net's 0.98778 accuracy, 0.91828 macro F1), which is interpreted as evidence that PEPE's explicit path scoring and TGAD's temporal attention contribute substantially beyond what the identity-graph topology and its four node attributes alone provide to a standard GNN, consistent with the ablation results in Table 6. No claim of comparative superiority over graph-based methods in general is made from this result: the three baselines use a minimal, four-attribute node representation and a fixed 2-layer configuration without dedicated hyperparameter tuning, so a more richly featured or more extensively tuned graph baseline could plausibly narrow this gap. Full training/evaluation code is available from the corresponding author upon reasonable request to allow independent verification and improvement of these baselines. Benchmarking against the five specifically surveyed methods, re-implemented on an identical identity graph and data split where feasible, remains identified as a priority for future work.

**(L4) AUC of 1.000 and data leakage.** The reported AUC is a support-weighted (micro-averaged) statistic driven primarily by the high-support Normal and Generic classes ( $F1 = 1.000$  and  $0.997$ , respectively), while minority classes (Analysis, Backdoor, Shellcode, and Worms) reach only  $0.85$ – $0.87$   $F1$ —consistent with the macro- $F1$  gap already reported ( $0.918$ ) and not indicative of uniformly perfect separability; per-class ROC curves are shown in Figure 3. Class balancing is applied prior to the stratified train-test split, and the two partitions remain disjointed, which mitigates but does not formally rule out all sources of leakage. Because balancing is applied before splitting rather than to the training fold alone, the reported test metrics characterize a class-balanced evaluation set rather than UNSW-NB15's natural, highly imbalanced distribution, which should be taken into account when comparing these figures to deployment-scale performance.

## 5. Conclusions

This paper proposes PEGraphSec-Net, a graph-theoretic framework for privilege-escalation-proxy threat detection in cloud-native identity systems, built around five complementary modules whose individual roles and equations are detailed in Section 3 and summarized in Figure 1. Rather than restate each module's description here, we summarize

only the evaluated outcome: the detection-side modules (IRGC, PEPE, and TGAD) are validated end-to-end against six baseline classifiers on held-out UNSW-NB15 data, while the containment-side modules (ACIE and MPRO) are validated through their measurable contribution to that same classification performance under ablation, with their live containment/policy-optimization behavior explicitly left for future validation (Section 4.9).

As reported in detail in Section 4, the model correctly classifies the large majority of test instances across all eleven UNSW-NB15 categories and outperforms multiple strong baseline models including CNN, LightGBM, XGBoost, Random Forest, SVM, and MLP, while the macro-averaged metrics introduced in Table 4 indicate remaining headroom on minority attack classes. Confusion matrix, ROC analysis, and convergence results further verified the robustness, stability, and generalization ability of the proposed approach. Interpretability analysis using SHAP confirmed that PEGraphSec-Net learns meaningful network traffic patterns for threat classification.

The results demonstrate that PEGraphSec-Net can detect the evaluated privilege-escalation-proxy threat patterns with low false alarm rates on the held-out, class-balanced test set. The priority directions below map directly onto the five scope conditions identified in Section 4.9 (L1–L5). Future work will explore real-time deployment of the framework in live cloud environments, extension to federated and multi-cloud settings—potentially building on privacy-preserving federated learning designs for distributed identity data [27]—and integration with zero-trust architecture frameworks for enhanced adaptive cyber defense. Priority directions include (i) validating PEGraphSec-Net on a native, modern cloud-IAM privilege-escalation dataset, rather than the decade-old UNSW-NB15 proxy mapping used here (L1); (ii) evaluating ACIE and MPRO through live or simulated containment actions with explicit reinforcement-learning metrics (cumulative reward, policy convergence, and operational disruption), rather than through classification-ablation alone (L2); (iii) benchmarking PEGraphSec-Net experimentally against the surveyed state-of-the-art methods and against re-implemented graph-based baselines (GAT, GraphSAGE, and temporal GNNs) operating on the identical identity graph [9,19,20] (L3); and (iv) exploring large language model-assisted interpretability and alert triage, following recent zero-shot LLM frameworks for classification and urgency scoring in adjacent domains [28], alongside generative-AI-aware risk mitigation strategies for the underlying detection pipeline [29].

**Author Contributions:** Conceptualization, K.R.A. and M.N.P.; methodology, K.R.A., M.N.P. and M.D.H.; software, M.N.P. and M.S.; validation, K.R.A. and M.R.R.; formal analysis, M.D.H. and M.S.; investigation, B.H. and M.R.R.; resources, K.R.A.; data curation, M.N.P. and B.H.; writing—original draft preparation, M.N.P. and M.D.H.; writing—review and editing, K.R.A. and M.R.R.; visualization, M.S. and B.H.; supervision, K.R.A. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** The dataset used in this study is publicly available at <https://www.kaggle.com/datasets/mrwellsdavid/unsw-nb15/data> (accessed on 18 May 2026).

**Acknowledgments:** The authors used AI-assisted writing tools during manuscript preparation, specifically large language model-based assistants (OpenAI ChatGPT-4o and Grammarly AI (2024 version)) for grammar checking, language improvement, and structural refinement of selected sections. These tools were used exclusively for linguistic editing of text written by the authors; they were not used to generate research data, create figures, produce novel scientific insights, conduct literature searches, or formulate the methodology or conclusions. All scientific content, including the model architecture, experimental design, data analysis, interpretation of results, and conclusions,

was developed, verified, and is solely the responsibility of the authors. No AI tools were used in the peer-review process.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## Abbreviations

The following abbreviations are used in this manuscript:

|                |                                             |
|----------------|---------------------------------------------|
| PEGraphSec-Net | Privilege Escalation Graph Security Network |
| IRGC           | Identity Relationship Graph Constructor     |
| PEPE           | Privilege Escalation Path Encoder           |
| TGAD           | Temporal Graph Attention Detection Layer    |
| ACIE           | Adaptive Containment and Isolation Engine   |
| MPRO           | Mitigation Policy Reinforcement Optimizer   |
| GNN            | Graph Neural Network                        |
| HGNN           | Heterogeneous Graph Neural Network          |
| GRU            | Gated Recurrent Unit                        |
| GAT            | Graph Attention Network                     |
| IAM            | Identity and Access Management              |
| SHAP           | SHapley Additive exPlanations               |
| AUC            | Area Under the Curve                        |
| ROC            | Receiver Operating Characteristic           |
| CNN            | Convolutional Neural Network                |
| MLP            | Multi-Layer Perceptron                      |
| SVM            | Support Vector Machine                      |
| GPU            | Graphics Processing Unit                    |

## References

- Indu, I.; Rubesh Anand, P.M.; Bhaskar, V. Identity and Access Management in Cloud Environment: Mechanisms and Challenges. *Eng. Sci. Technol. Int. J.* **2018**, *21*, 574–588. <https://doi.org/10.1016/j.jestch.2018.05.010>.
- Saqib, M.; Mehta, D.; Yashu, F.; Malhotra, S. Adaptive Security Policy Management in Cloud Environments Using Reinforcement Learning. *arXiv* **2025**, arXiv:2505.08837.
- Devaraju, P.; Devarapalli, S.; Tuniki, R.R.; Kamatala, S. Secure and Adaptive Federated Learning Pipelines: A Framework for Multi-Tenant Enterprise Data Systems. In *Proceedings of the 2025 International Conference on Computing Technologies (ICOCT)*, Bengaluru, India; IEEE: Piscataway, NJ, USA, 2025; pp. 1–7. <https://doi.org/10.1109/ICOCT64433.2025.11118425>.
- Madireddy, V.T. Graph Neural Network-Based Adaptive Threat Detection for Cloud Identity and Access Management Logs. *arXiv* **2025**, arXiv:2512.10280. <https://doi.org/10.48550/arXiv.2512.10280>.
- Pomsathit, A. Adaptive Detection of Advanced Persistent Threats (APT) with Graph Neural Networks and Rehearsal-Based Continual Learning on Wazuh EDR Telemetry. *IEEE Access* **2025**, *13*, 212973–212982. <https://doi.org/10.1109/ACCESS.2025.3639270>.
- Li, X.; Li, X.; Jia, J.; Li, L.; Yuan, J.; Gao, Y.; Yu, S. A High Accuracy and Adaptive Anomaly Detection Model with Dual-Domain Graph Convolutional Network for Insider Threat Detection. *IEEE Trans. Inf. Forensics Secur.* **2023**, *18*, 1638–1652. <https://doi.org/10.1109/TIFS.2023.3245413>.
- Mansour Bahar, A.A.; Ferrahi, K.S.; Messai, M.L.; Seba, H.; Amrouche, K. CONTINUUM: Detecting APT Attacks through Spatial-Temporal Graph Neural Networks. *arXiv* **2025**, arXiv:2501.02981. <https://doi.org/10.48550/arXiv.2501.02981>.
- Canonico, R.; Lista, F.; Navarro, A.; Sperli, G.; Vignali, A. Threat Detection in Reconfigurable Cyber–Physical Systems through Spatio-Temporal Anomaly Detection Using Graph Attention Network. *Comput. Secur.* **2025**, *156*, 104509. <https://doi.org/10.1016/j.cose.2025.104509>.
- Rekik, S.; Mehmood, S. A Hybrid Graph Neural Network and Neural ODE Model to Intrusion Detection in Dynamic Network Topologies. *IEEE Access* **2025**, *13*, 198201–198227. <https://doi.org/10.1109/ACCESS.2025.3635385>.
- Iannucci, S.; Shafik, W.; Matovu, R. AI-Driven Threat Intelligence and Autonomous Response in Cloud-Native Security Operations. *IEEE Access* **2025**, *13*, 24871–24891. <https://doi.org/10.1109/ACCESS.2025.3527684>.
- Zeng, Y.; Liu, H.; Cheng, S.; Lv, J. Graph-Based Access Control Policy Verification for Microservice Architectures in Cloud Environments. *Comput. Secur.* **2025**, *148*, 104138. <https://doi.org/10.1016/j.cose.2024.104138>.

12. Morić, Z.; Dakić, V.; Regvart, D.; Redžepagić, J. AMNDA: An Adaptive Multi-Layer, Lifecycle-Aware Defense Architecture for Multi-Stage Cyberattacks with Azure-Based Validation. *Electronics* **2026**, *15*, 1939. <https://doi.org/10.3390/electronics15091939>.
13. Zhang, Z.; Chen, D.; Yang, X.; Mao, S.; Liu, Y. Digital Twins in Intelligent Transportation and Communication Systems: A Survey. 2026. Available online: <https://zenodo.org/records/19136071> (accessed on 23 July 2026).
14. Ogunmolu, A.M.; Olaniyi, O.M.; Ejoh, C.I.; Balogun, A.Y.; Oyekunle, S.M. Secure-by-Containment Architectures for Agentic AI in OT/ICS Environments. *Asian J. Res. Comput. Sci.* **2026**, *19*, 1–23. <https://doi.org/10.9734/ajrcos/2026/v19i3833>.
15. Prabhu, S.; Sahoo, S.K.; Choudhury, T. Identity and Access Management Threats in Multi-Cloud Environments: A Systematic Review. *Comput. Secur.* **2024**, *138*, 103670. <https://doi.org/10.1016/j.cose.2023.103670>.
16. Cai, L.; Chen, Z.; Luo, C.; Gui, J.; Noseworthy, J.; Song, D.; Chen, H. Structural Temporal Graph Neural Networks for Anomaly Detection in Dynamic Graphs. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management*; Association for Computing Machinery: New York, NY, USA, 2024. <https://doi.org/10.1145/3459637.3481955>.
17. Nair, V.; Sharma, P.; Gupta, M. Reinforcement Learning for Adaptive Intrusion Response in Cloud Microservice Architectures. *J. Netw. Comput. Appl.* **2024**, *221*, 103768.
18. M K, V.P.; S, S.; H, A.G.; Aggarwal, M.; VS, S.S. Advanced Persistent Threat Detection through Sequential Analysis of Network Patterns with Graph Based Learning Approach. *Sci. Rep.* **2026**, *16*, 19998. <https://doi.org/10.1038/s41598-026-42756-w>.
19. Lan, K.; Li, G.; Huang, W.; Li, J.; Yu, Y. LLM-HGAN: LLM-Enhanced Heterogeneous Graph Attention Networks for Advanced Persistent Threat Detection. *IEEE Trans. Netw. Sci. Eng.* **2026**, *13*, 8892–8910. <https://doi.org/10.1109/TNSE.2026.3687554>.
20. Kou, L.; Pan, X.; Dong, G.; Wang, M.; Miao, C.; Zhang, J.; Duan, P. Dynamic Adaptive Aggregation and Feature Pyramid Network Enhanced GraphSAGE for Advanced Persistent Threat Detection in Next-Generation Communication Networks. *IEEE Trans. Netw. Serv. Manag.* **2026**, *23*, 2712–2727. <https://doi.org/10.1109/TNSM.2026.3660650>.
21. Shabani, N.; Beheshti, A.; Qi, Y.; Haghighi, V.; Foo, J.; Wu, J. STGS: Spatio-temporal Graph Sparsification Using Reinforcement Learning. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management (CIKM '25)*; Association for Computing Machinery: New York, NY, USA, 2025; pp. 2546–2555. <https://doi.org/10.1145/3746252.3761107>.
22. Zhou, J.; Shen, D.; Guo, Y.; Zhao, Y.; Zhang, H. Enhancing Knowledge Graph Recommendations through Deep Reinforcement Learning. *Sci. Rep.* **2026**, *16*, 3076. <https://doi.org/10.1038/s41598-025-31109-8>.
23. Wang, R.; Zhang, J.; Wang, X.; He, H.; Zou, Y. Multi-Modal and Multi-Agent Reinforcement Learning Framework for Urban Traffic Flow Prediction and Signal Control Optimization. *Sci. Rep.* **2026**, *16*, 7612. <https://doi.org/10.1038/s41598-026-37722-5>.
24. Shivahare, B.; Singh, G.; Naz, R.; Yadav, S.A.; Mathivanan, S.K. AI-Driven Hybrid Framework for Enhanced Pest Detection and Resource Optimization Using Graph Networks and Deep Reinforcement Learning. *Sci. Rep.* **2026**, *16*, 18131. <https://doi.org/10.1038/s41598-026-47987-5>.
25. Zhang, K.; Yang, X.; Li, H. Intelligent Decision-Making for Mine Ventilation Systems Based on Graph Neural Network and Deep Reinforcement Learning Fusion. *Sci. Rep.* **2026**, *16*, 6704. <https://doi.org/10.1038/s41598-026-37347-8>.
26. Adjei, P.K.; Zhiguang, Q.; Obiri, I.A.; Badije, A.; Cobblah, C.N.A.; Alqahtani, A.; Gu, Y.H.; Al-Antari, M.A. A Graph Attention Network-Based Multi-Agent Reinforcement Learning Framework for Robust Detection of Smart Contract Vulnerabilities. *Sci. Rep.* **2025**, *15*, 29810. <https://doi.org/10.1038/s41598-025-14032-w>.
27. Wang, R.; Lai, J.; Li, X.; He, D.; Khan, M.K. RPIFL: Reliable and Privacy-Preserving Federated Learning for the Internet of Things. *J. Netw. Comput. Appl.* **2024**, *221*, 103768. <https://doi.org/10.1016/j.jnca.2023.103768>.
28. Rajkumar, S.C.; Yuvasini, D.; Selvarajan, S.; Sivakumar, N.R. A Zero-Shot LLM Framework for Multimodal Grievance Classification, Urgency Scoring, and Abuse Detection in Civic Feedback Systems. *Sci. Rep.* **2026**, *16*, 2299. <https://doi.org/10.1038/s41598-025-32079-7>.
29. Al-Hashimi, H.A. A Generative AI Cybersecurity Risks Mitigation Model for Code Generation: Using ANN-ISM Hybrid Approach. *Sci. Rep.* **2026**, *16*, 4239. <https://doi.org/10.1038/s41598-025-34350-3>.
30. Moustafa, N.; Slay, J. UNSW-NB15: A Comprehensive Data Set for Network Intrusion Detection Systems. In *Proceedings of the 2015 Military Communications and Information Systems Conference (MilCIS)*; IEEE: Piscataway, NJ, USA, 2015. <https://doi.org/10.1109/MilCIS.2015.7348942>.

**Disclaimer/Publisher's Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.