Noname manuscript No.
(will be inserted by the editor)

FedSymGraph: A Hybrid Federated Graph Learning Framework with Symbolic–LLM Reasoning for Explainable and Privacy-Preserving Intrusion Detection

Ahmed Saidi

Received: / Accepted:

Abstract The rapid growth of connected devices and complex network infrastructure has amplified the scale and sophistication of cyber threats. Traditional intrusion detection systems (IDS), even those that leverage deep learning or large language models (LLMs), remain vulnerable to unseen attacks, suffer from centralization bottlenecks, and offer limited interpretability. To address these challenges, this paper proposes FedSymGraph, a hybrid federated graph learning framework that integrates graph neural networks (GNNs), symbolic reasoning, and LLM-based explanation to enable collaborative, privacy-preserving, and explainable intrusion detection.

FedSymGraph models network traffic and system events as dynamic heterogeneous graphs, capturing structural and temporal dependencies to detect anomalous behaviors beyond isolated features. A federated training design allows multiple domains—including enterprises, IoT gateways, and cloud platforms—to jointly train models without sharing raw data, ensuring compliance with privacy. Symbolic reasoning based on the MITRE ATT&CK framework, combined with an LLM-based explanation layer, provides interpretable, human-readable justifications for detections. An adaptive reinforcement controller optimises detection thresholds and communication overhead in real time.

Experimental evaluations on the CIC-IDS2017 and DARPA Transparent Computing datasets demonstrate that FedSymGraph achieves superior accuracy, robustness, and interpretability compared to state-of-the-art IDS approaches, while maintaining strong privacy guaranties and efficient communication. The framework offers a scalable, collaborative, and reliable solution for modern network security environments.

Keywords Federated learning · Graph neural network · Symbolic reasoning · Large language model · Intrusion detection · Privacy preservation · Explainable AI

Ahmed Saidi
Research Center for Scientific and Technical Information (CERIST), 16000 Algiers, Algeria
E-mail: a.saidi@cerist.dz

1 Introduction

The exponential growth of connected infrastructures and networks has generated vast, heterogeneous volumes of network data. While this connectivity enables advanced services and intelligent systems, it also exposes networks to sophisticated cyber threats, including multi-stage attacks, lateral movement, and zero-day exploits. Intrusion Detection Systems (IDS) are essential to maintain operational integrity, trust, and security in distributed

environments. Conventional IDS—whether rule-based or machine learning-based—struggle to detect novel attacks, preserve privacy, and provide interpretable explanations for their alerts.

Recent advances in deep learning and large language models (LLMs) offer improved detection capabilities by automatically inferring complex patterns in network traffic. However, these models typically rely on centralized training, requiring access to sensitive data and high computational resources. Centralized approaches pose challenges for privacy compliance, scalability, and explainability, limiting their use in distributed and regulated environments.

To address these limitations, this work introduces **FedSymGraph**, a hybrid framework that integrates graph neural networks (GNNs), symbolic reasoning, and federated learning. GNNs are introduced to capture the topological structure of networked systems and model complex dependencies between hosts, services, and events. Unlike conventional feature-based approaches, GNNs leverage relational and temporal information from dynamic heterogeneous graphs, enabling accurate detection of multi-stage and coordinated attacks.

A symbolic–LLM reasoning layer provides interpretable and causally grounded explanations for detected anomalies. Although GNNs capture structural patterns, their output remains opaque and difficult to interpret. By combining symbolic reasoning grounded in MITRE ATT&CK with LLM-based natural language explanations, the framework produces alerts that are traceable, understandable, and actionable.

Federated learning underpins collaborative model training, allowing multiple domains—including enterprises, IoT gateways, and cloud platforms—to jointly improve detection without sharing raw network data. This preserves privacy, ensures regulatory compliance, and leverages diverse data sources. By combining GNN-based structural modeling, symbolic–LLM reasoning, and federated protocols, FedSymGraph delivers a scalable, privacy-preserving, and interpretable IDS solution suitable for modern distributed networks.

Motivating Scenario:

Consider a distributed enterprise consisting of multiple sub-networks: a headquarters data center, several branch offices, and IoT-enabled manufacturing sites. Each domain continuously collects network traffic data and system audit logs. Traditional intrusion detection systems (IDS) require central aggregation of this information, which violates privacy and increases the risk of exposure of sensitive data. Furthermore, when a novel attack occurs in one domain—such as an SSH brute-force attempt or lateral privilege escalation—centralized systems cannot promptly propagate defensive knowledge to other domains.

FedSymGraph enables each domain to function as an autonomous learner that contributes knowledge to a shared global model without revealing sensitive data. Network traffic is represented as a temporal heterogeneous graph, where entities—including hosts, users, ports, and processes—form nodes, and their interactions form edges. Local Graph Neural Networks (GNNs) learn rich representations of normal and abnormal behaviors, capturing both relational and temporal dependencies. Encrypted model updates are aggregated by a global federated server, ensuring privacy and regulatory compliance.

Detected anomalies are further analyzed using a symbolic reasoning engine to identify the corresponding MITRE ATT&CK techniques. A Large Language Model (LLM) layer translates these results into interpretable, human-readable reports, providing actionable insights for security analysts.

This framework ensures that intrusion detection remains collaborative, interpretable, privacy-preserving, and adaptable to previously unseen attacks across distributed network environments.

Challenges:
Motivated by the above observations, this paper addresses the following key challenges:

– **Data Privacy:**
Ensuring the confidentiality of sensitive network data is essential. Even within federated learning, participants can infer information from shared model updates. The framework integrates mech-

anisms such as differential privacy and secure aggregation to prevent data leakage while maintaining the utility of the model.

– **Non-IID and Heterogeneous Data:**
Network traffic across domains is non-identically distributed due to variations in topology, devices, and operational patterns. The system must ensure robust convergence and consistent performance under heterogeneous conditions.

– **Explainability and Semantic Validity:**
Deep models often operate as black boxes. LLM-based explanations may be fluent but lack causal reasoning. The proposed framework combines symbolic reasoning with LLM interpretation to produce logical and semantically grounded explanations.

– **Communication Efficiency:**
Large graph-based model updates can create significant network overhead. Efficient aggregation and compression strategies are required to maintain scalability and reduce communication cost.

– **Client and Fog Heterogeneity:**
Domains differ in computational resources, network reliability, and data quality. The framework must accommodate this heterogeneity to ensure stable global model convergence.

– **Scalability and Robustness:**
As the number of participants increases, maintaining performance, privacy, and reliability becomes complex. Scalable aggregation and adaptive coordination mechanisms are incorporated to ensure robustness under distributed conditions.

By tackling these challenges, the proposed framework aims to establish an interpretable, efficient, and privacy-preserving federated graph learning system suitable for diverse and large-scale network environments.

Contributions:

The main contributions of this paper are summarized as follows:

– **Hybrid Federated Graph Learning Architecture:**
We propose **FedSymGraph**, a hybrid federated graph learning framework that integrates Graph Neural Networks (GNNs), symbolic reasoning, and Large Language Model (LLM) interpretation to achieve privacy-preserving and explainable intrusion detection.

– **Graph-based Representation Learning:**
Network behaviors are modeled as dynamic temporal graphs that capture both topological and causal relationships between entities. This representation improves the system's ability to detect zero-day and context-dependent attacks.

– **Privacy-Preserving Collaboration:**
The framework employs differential privacy and secure aggregation to ensure the confidentiality of model updates and compliance with data protection regulations, without requiring centralized data sharing.

– **Symbolic–LLM Explainability:**
We design a hybrid reasoning engine that links GNN-detected anomalies to the MITRE ATT&CK knowledge base and validates LLM-generated explanations for semantic and causal consistency.

– **Adaptive Reinforcement Control:**
A reinforcement learning controller dynamically adjusts privacy noise levels, communication intervals, and anomaly detection thresholds based on real-time feedback to optimize detection accuracy and efficiency.

– **Comprehensive Evaluation:**
Extensive experiments conducted on the CIC-IDS2017 and DARPA Transparent Computing datasets demonstrate that FedSymGraph achieves superior detection accuracy, lower false-positive rates, and improved interpretability compared to existing federated and neural baselines.

By integrating neural, symbolic, and federated intelligence, FedSymGraph establishes a robust, explainable, and privacy-preserving foundation for next-generation intrusion detection systems.

Paper Organization:

2 Related Work

2.1 Comparison with Related Work and Discussion

3 Background

3.1 Notations

3.2 Preliminaries

4 System Model and Framework

5 System Model and Framework

In this section, we provide a detailed description of the proposed FedSymGraph architecture and its operational workflow. We also define the threat model and outline the security assumptions that guide the design of the system .

5.1 System Model

Figure **??** illustrates the architecture and operation of the proposed hybrid federated intrusion detection framework. FedSymGraph is composed of four principal components: a Global Coordinator, distributed Domain Nodes, a Symbolic Reasoning Engine, and an LLM-based Explanation Layer.

– **Global Coordinator:**
The Global Coordinator serves as the trusted orchestrator responsible for system initialization, global aggregation, and model distribution. It performs cryptographic setup, manages key exchange, and enforces privacy parameters such as differential privacy budgets. The coordinator aggregates encrypted model updates from distributed domains and disseminates the global model to participating nodes.

– **Domain Nodes:**
Domain Nodes represent organizational networks such as enterprises, IoT infrastructures, or cloud environments. Each domain independently trains a local Graph Neural Network (GNN) on its network telemetry, represented as a temporal communication graph. Only encrypted or differentially private model updates are shared with the Global Coordinator, ensuring that raw network traffic never leaves the local environment.

– **Symbolic Reasoning Engine:**
This engine maintains a knowledge base aligned with the MITRE ATT&CK framework. It interprets detected anomalies from local or aggregated models and maps them to corresponding tactics and techniques. The symbolic reasoning layer ensures semantic grounding and causal explanation for every detection decision.

– **LLM Explanation Layer:**
A fine-tuned large language model (LLM) generates human-readable explanations of detected anomalies. It translates the outcomes symbolic reasoning into contextual narratives that support analyst understanding. The LLM output is validated by the symbolic layer to ensure accuracy and prevent hallucinated interpretations.

The operation of FedSymGraph follows an iterative process. Initially, each Domain Node constructs a dynamic network graph from local traffic, trains a GNN model, and applies differential privacy to its gradients. Encrypted model updates are transmitted to the Global Coordinator, which performs secure aggregation and updates the global model. Detected anomalies are processed by the Symbolic Reasoning Engine and the LLM layer for interpretation. This process repeats until convergence, ensuring collaborative and interpretable intrusion detection across all domains.

The integration of federated learning, graph-based modeling, and symbolic–LLM reasoning ensures privacy, interpretability, and scalability

for distributed intrusion detection.

5.2 Threat Model

The FedSymGraph environment operates in a partially trusted environment where participants collaborate under privacy-preserving constraints. The following adversarial threats are considered within the scope:

– **Inference Attacks:**
Adversaries may attempt to infer sensitive network information from shared model updates, such as IP correlations or device behavior. Differential privacy mechanisms prevent gradient leakage and protect against membership inference attacks.

– **Model Poisoning:**
Compromised participants may inject manipulated updates to bias the global model or degrade detection accuracy. Robust aggregation techniques mitigate the effect of outliers or inconsistent updates.

– **Information Leakage:**
Eavesdroppers may intercept communication between nodes and the Global Coordinator. Encrypted transmission and authenticated channels ensure data confidentiality and integrity.

– **Explanation Manipulation:**
An adversary could attempt to induce misleading outputs from the LLM-based explanation layer. Symbolic reasoning validation prevents unverified or false explanations from being accepted.

The threat model assumes a semi-honest environment where participants follow the training protocol but may attempt to infer information from model updates. External attackers can intercept network communications, but cannot compromise cryptographic primitives.

5.3 Security Assumptions and Requirements

The FedSymGraph framework operates under the following assumptions and design requirements:

– **Global Coordinator:**
Assumed to be trusted and responsible for initialization, encryption key management, and global aggregation. It ensures compliance with data privacy regulations, such as GDPR and enforces differential privacy across the federation.

– **Domain Nodes:**
Considered honest-but-curious. They perform local training correctly, but may attempt to extract insights from aggregated models. Secure aggregation and differential privacy prevent the leakage of sensitive information.

– **Symbolic Reasoning Engine:**
Operates as a semi-trusted component that interprets and validates anomalies. It accesses only aggregated indicators, not raw network data, ensuring analytical transparency without compromising privacy.

– **LLM Explanation Layer:**
The LLM functions as an auxiliary interpreter. It does not access raw datasets or gradients; instead, it receives symbolic summaries validated by the reasoning engine, preserving semantic integrity.

– **Communication Channels:**
All communication between Domain Nodes and the Global Coordinator is encrypted and authenticated. Secure transport protocols (e.g., TLS) protect against interception or tampering.

Security Model Coverage:
FedSymGraph satisfies the following security and privacy requirements:

– **Data Confidentiality:** Sensitive network data remain within local domains. Only differentially private and encrypted updates are exchanged.

– **Data Integrity:** Robust aggregation ensures that only consistent and validated updates

contribute to the global model.

– **Authentication:** Digital signatures verify the participant's identity and update authenticity.

– **Privacy Preservation:** Differential privacy prevents reconstruction of local data from model gradients.

– **Explainability Integrity:** Symbolic reasoning verifies all explanations produced by the LLM to prevent hallucination and maintain causal traceability.

– **Scalability and Availability:** The federated architecture and parallel aggregation across domains ensure continued operation under distributed and large-scale conditions.

These assumptions and requirements collectively ensure that FedSymGraph achieves confidentiality, integrity, interpretability, and scalability in distributed intrusion detection.

5.4 Proposed Framework: FedSymGraph Architecture

The proposed FedSymGraph framework integrates federated graph learning, differential privacy, secure aggregation, symbolic reasoning, and LLM-based explanation. The goal is to provide privacy-preserving, interpretable, and scalable intrusion detection across distributed network environments. Each component addresses challenges related to data confidentiality, heterogeneous traffic, efficient communication, and semantic consistency in explanations. The following subsection describes the architecture and its main processes in detail.

5.5 System Initialization

In the initialization phase, the Trusted Aggregator (TA) establishes the foundation of the FedSymGraph system. The TA defines the set of participating organizations U, registers each domain, and creates the initial global graph model that will be distributed during the first training round. The TA also determines the privacy budget, secure aggregation parameters, and the reward structure for the reinforcement controller. These settings ensure that training begins under consistent privacy and operational constraints. The overall process is outlined in Algorithm **??**..

Each domain receives a unique identifier and a set of configuration parameters that specify local graph construction rules, supported feature types, and model update formats. This ensures that all participants generate compatible graph representations and federated updates during training. The initialization can be summarized as:

$$\text{Init}(U, \Theta_0, \varepsilon, \sigma) \rightarrow (\text{GlobalModel}_0, \text{Config})$$

where U is the set of domains, Θ_0 is the initial model, ε is the privacy budget, and σ is the differential privacy noise scale.

During this phase, the TA also verifies that each domain meets the minimum reliability requirements. Domains must demonstrate stable connectivity, sufficient computational capacity for GNN training, and compliance with model update formatting rules. This prevents invalid or incompatible contributions in later rounds.

To avoid unnecessary communication overhead during early rounds, the TA sets a baseline communication schedule and distributes it to all participants. This includes the expected interval between local updates, the maximum allowable size of a model message, and the initial anomaly detection threshold used across domains.

The initialization phase ensures that all organizations start training with a unified model, consistent operational rules, and a predefined privacy and security specification. This creates the structural foundation for cross-domain training and for the integration of graph learning, symbolic reasoning, and LLM-based interpretation in the later stages of the FedSymGraph workflow.

Algorithm 1 System Initialization for FedSymGraph

Input: Security parameter k, set of domains U, attribute universe A
Output: Public key PK, master secret key MSK, attribute-based keys for all domains

$(PK, MSK) \leftarrow Setup(1^k, U, A)$
TA stores MSK securely and distributes PK to all domains
for each domain $D_i \in U$ **do**
    Domain solves PoW challenge: find *nonce* such that
    $H(nonce \parallel ID_i \parallel PK) \leq Target$
    if PoW valid **then**
        TA generates attribute key $SK_i \leftarrow KeyGen(MSK, Attr_i)$
        Register domain D_i
    else
        Reject registration request

5.6 Local Graph Learning

During the local learning phase, each participating domain constructs temporal graphs from its network or system data and trains a Graph Neural Network (GNN) to model normal and abnormal behavior. The objective is to extract structural and temporal patterns without transferring raw data outside the domain, ensuring privacy by design. The workflow is illustrated in Algorithm 2.

Each organization converts raw logs into a sequence of time-indexed graphs:

$$G_t = (V_t, E_t, A_t)$$

where

- V_t is the set of nodes (IP addresses, users, processes)
- E_t is the set of edges represent interactions (connections, read, write, execute)
- A_t contains node and edge attributes at time t

A domain D_k maintains a series of such graphs:

$$G_k = \{G_t^k\}_{t=1}^T$$

capturing the evolution of behaviour over time. This temporal representation allows the system to detect subtle drifts that may indicate malicious activity.

Example

Suppose a server has the following log entries at time t:

Source	Destination	Action
192.168.1.10	192.168.1.20	connect
192.168.1.20	192.168.1.30	read
192.168.1.30	192.168.1.10	execute

This can be represented as a temporal graph:

- Nodes: v_1 = 192.168.1.10, v_2 = 192.168.1.20, v_3 = 192.168.1.30
- Edges: $(v_1 \to v_2), (v_2 \to v_3), (v_3 \to v_1)$
- Node features: $v_1 = [0.7, 0.4, 0.9]$ (traffic volume, active connections, protocol activity), $v_2 = [0.3, 0.6, 0.2]$, $v_3 = [0.5, 0.2, 0.8]$

Each domain trains a GNN using message passing layers of the form:

$$h_v^{(l+1)} = \sigma\Big(W^{(l)} \cdot \mathrm{AGG}\big(\{h_u^{(l)} \mid u \in N(v)\} \cup \{h_v^{(l)}\}\big)\Big)$$

where

- $h_v^{(l)}$ is the embedding of node v at layer l
- $N(v)$ is in the neighborhood of v
- $\mathrm{AGG}(\cdot)$ is a permutation-invariant aggregation operator such as mean, sum, or attention
- $W^{(l)}$ is a trainable weight matrix
- $\sigma(\cdot)$ is an activation function

For example, node v_1 updates its embedding by aggregating its own features with those of v_2.

Algorithm 2 Local Graph Learning at Domain D_k

Input: Local logs $\mathcal{L}_k$, initial model w_t, learning rate η, epochs E
Output: Noisy protected model update $\tilde{w}_k$

1: Convert logs $\mathcal{L}_k$ into temporal graphs $G_k = \{G_t^k\}_{t=1}^T$
2: **for** each graph snapshot G_t^k **do**
3: Perform GNN message passing:

$$h_v^{(l+1)} = \sigma\big(W^{(l)} \cdot AGG(\{h_u^{(l)} : u \in N(v)\} \cup h_v^{(l)})\big)$$

4: Compute self-supervised loss:

$$L_k = L_{rec} + \lambda_1 L_{contrast} + \lambda_2 L_{context}$$

5: Update parameters:

$$w_k^{t+1} = w_t - \eta \nabla L_k(w_t)$$

6: Add DP noise:

$$\tilde{w}_k = w_k^{t+1} + \mathcal{N}(0, \sigma^2 I)$$

7: **return** $\tilde{w}_k$

To capture structural and temporal consistency, each domain optimizes a self-supervised objective:

$$L_k = L_{\text{rec}} + \lambda_1 L_{\text{contrast}} + \lambda_2 L_{\text{context}}$$

where

- L_{rec} reconstructs node features or adjacency structure
- L_{contrast} distinguishes between positive and negative graph pairs
- L_{context} predicts contextual behaviour from historical snapshots

The contrastive term is defined as:

$$L_{\text{contrast}} = -\sum_{v \in V_t} \log \frac{\exp(\text{sim}(h_v, h_v^+))}{\exp(\text{sim}(h_v, h_v^+)) + \sum_{h_v^-} \exp(\text{sim}(h_v, h_v^-))}$$

where h_v^+ and h_v^- denote positive and negative samples derived from temporal augmentations.

After training, anomaly scores are computed by comparing embeddings with their reconstructions. For example, if the reconstruction error for v_1 is $s_{v_1} = 0.15$ and the threshold $\tau = 0.1$, node v_1 is marked as anomalous:

$$s_v = \|h_v - \hat{h}_v\|_2^2$$

Before any model update leaves the local domain, differential privacy is applied:

$$\tilde{w}_k = w_k + \mathcal{N}(0, \sigma^2 I)$$

ensuring that no single record or behaviour can be reconstructed from gradients or parameters.

Finally, each domain sends the protected update $\tilde{w}_k$ and anomaly metadata to the federated aggregator. This enables cross-domain learning while maintaining strict confidentiality and preserving the integrity of local datasets.

5.7 Federated Aggregation and Privacy Protection

After each domain completes local graph learning, the system proceeds to federated aggregation. The objective is to combine the protected model updates from all participating domains without exposing raw data or sensitive information contained in local computations. FedSymGraph achieves this through a combination of differential privacy, encrypted model submission, trust-weighted aggregation, and anomaly-aware filtering. The complete aggregation pipeline is summarized in Algorithm **??**.

Algorithm 3 Federated Aggregation with Differential Privacy and Secure Aggregation

Input: Protected client updates $\{\tilde{w}_k\}_{k=1}^K$, trust coefficients $\{q_k\}$
Output: Updated global model w_{t+1}

1: **for** each received update $\tilde{w}_k$ **do**
2: Verify validity and consistency
3: Apply trust weighting:

$$\omega_k = \frac{n_k q_k}{\sum_j n_j q_j}$$

4: Compute aggregated model:

$$w_{t+1} = \sum_{k=1}^{K} \omega_k \tilde{w}_k$$

5: Distribute w_{t+1} to all domains

Each client D_k maintains a local model w_k^t at training round t. Following local optimisation, the model is updated according to:

$$w_k^{t+1} = w_k^t - \eta \nabla L_k(w_k^t)$$

where η is the learning rate and L_k is the local self-supervised loss defined previously. Before transmission, the domain applies differential privacy to prevent gradient leakage:

$$\tilde{w}_k^{t+1} = w_k^{t+1} + \mathcal{N}(0, \sigma^2 I)$$

Here, σ represents the privacy noise scale that determines the privacy–utility trade-off, and I is the identity covariance matrix. A larger σ provides stronger privacy at the cost of reduced accuracy.

To ensure secure transmission, each update is encrypted using a symmetric or homomorphic encryption mechanism, depending on deployment constraints. The generic encryption process is denoted as:

$$c_k^{t+1} = \mathrm{Enc}_{K_k}(\tilde{w}_k^{t+1})$$

where K_k is a session-specific encryption key.

Upon receiving encrypted updates from all participating clients, the federated aggregator performs a secure aggregation protocol to obtain the global model. The aggregator never accesses individual plaintext updates; instead, it computes an encrypted weighted sum:

$$\mathrm{Enc}(w^{t+1}) = \frac{\sum_{k=1}^{K} n_k q_k \, \mathrm{Enc}(\tilde{w}_k^{t+1})}{\sum_{j=1}^{K} n_j q_j}$$

where:

- n_k is the number of samples (or graph nodes) in domain D_k
- q_k is a trust coefficient assigned to each domain
- q_k is dynamically updated based on reliability:

$$q_k = \alpha q_k^{\mathrm{prev}} + (1 - \alpha)\mathbf{1}[\text{consistent updates}]$$

A client is considered "consistent" when its update is within a tolerance range of the robust gradient mean:

$$\|\tilde{w}_k^{t+1} - \bar{w}^t\|_2 \leq \gamma$$

This rule filters out anomalous or poisoned updates and contributes to Byzantine resilience.

After aggregation, the encrypted global model is decrypted using the shared protocol keys:

$$w^{t+1} = \mathrm{Dec}(\mathrm{Enc}(w^{t+1}))$$

The global update is then distributed back to all domains. To ensure compliance with fine-grained access policies, the model is encrypted under CP-ABE:

$$C = \text{Enc}_{\text{CP-ABE}}(w^{t+1}, \text{Policy})$$

Only clients with attribute sets satisfying the policy can recover the model:

$$\text{Dec}_{\text{CP-ABE}}(C, SK_k) = w^{t+1} \quad \text{if Attr}(D_k) \models \text{Policy}$$

This guarantees that sensitive model parameters remain accessible only to authorised organisations.

The federated aggregation loop continues until the global model converges or reaches a plateau in performance. During each iteration, FedSymGraph ensures that:

- private data never leave local domains
- shared updates reveal no identifiable information
- malicious updates are detected and down-weighted
- only trusted and authorised nodes participate in the training loop

Through this combination of robust aggregation, privacy protection, and trust-aware weighting, FedSymGraph maintains high detection performance while strictly preserving confidentiality and system integrity.

5.8 Symbolic Reasoning Engine

The symbolic reasoning engine interprets anomalies detected by the GNN. It maps structural and temporal deviations to high-level attack tactics using a predefined rule base. The reasoning workflow is outlined in Algorithm 4. This layer introduces explainable, logic-grounded inference that complements the neural detection process. The symbolic engine enhances transparency and enables analysts to understand not only what anomaly occurred but why. The following paragraphs describe the rule representation, matching procedure, and generation of reasoning outputs.

5.8.1 Knowledge Base and Logical Rules

The symbolic engine uses a structured collection of rules derived from MITRE ATT&CK. Each rule corresponds to a specific attack technique and maps behavioural predicates to a semantic label. A rule r_i takes the general form:

$$r_i : (p_1 \wedge p_2 \wedge \cdots \wedge p_m) \Rightarrow a_i$$

where p_j denotes an event predicate extracted from graphs, and a_i represents the inferred attack type. The predicates reflect low-level observations such as repeated failures, privilege escalation attempts, or outbound data transfer.

A representative rule for credential abuse is:

$$r_{\text{T1110}}(\text{AuthFail} > 3 \wedge \text{AuthSuccess} \wedge \text{OutboundConn}) \Rightarrow \text{BruteForce}$$

Each rule is stored with metadata including severity, phase of operation, and temporal constraints.

Example – SSH Brute Force Consider event predicates extracted from system logs:

$$\text{AuthFail} > 5, \quad \text{AuthSuccess} = \text{true}, \quad \text{OutboundConn} = \text{true}$$

The corresponding rule:

$$r_{T1110} : (\text{AuthFail} > 5 \wedge \text{AuthSuccess} \wedge \text{OutboundConn}) \Rightarrow \text{BruteForce}$$

When these predicates hold simultaneously in the subgraph associated with the anomalous IP, the reasoning engine infers:

$$\text{AttackID} = T1110, \quad \text{Confidence} = 0.87$$

Algorithm 4 Symbolic Reasoning using MITRE-based Rule Evaluation

Input: Anomalous subgraph S, rule base R
Output: Symbolic inference tuple $(AttackID, Confidence, Path)$

```
Extract event predicates from S
for each rule r_i : (p_1 ∧ p_2 ··· ∧ p_m) ⇒ a_i in R do
    if {p_1, ..., p_m} ⊆ Events(S) then
        Compute confidence:
```

$$C_i = \frac{|Matched\ Predicates|}{m}$$

```
        Determine causal path using temporal ordering of edges
        return (a_i, C_i, Path_i)
return (None, 0, ∅)
```

5.8.2 Subgraph Extraction and Pattern Matching

When the GNN flags a node or subgraph as anomalous, the symbolic engine extracts the associated structure. The extracted subgraph

$$S = (V_s, E_s, t_s)$$

preserves the set of nodes, edges, and timestamps involved in the anomaly. Pattern matching evaluates whether S satisfies the antecedent of a rule. The matching operator is defined as:

$$\text{Match}(S, r_i) = \begin{cases} 1, & \text{if } S \models (p_1 \wedge \cdots \wedge p_m) \\ 0, & \text{otherwise} \end{cases}$$

Graph isomorphism checks are used to align the structure of S with the rule pattern. Temporal operators such as ordering, duration, and interval compliance are also evaluated:

$$t(p_j) \leq t(p_{j+1}), \quad \Delta t \leq \theta_i$$

where θ_i is the maximum temporal window for the rule.

Example – Suspicious Lateral Movement
Suppose the anomalous subgraph is:

- Node A → Node B (SMB File Read)
- Node B → Domain Controller (Kerberos Request)
- Node A performs privilege escalation shortly after

Rule template:

$$(\text{FileRead} \wedge \text{KerberosReq} \wedge \text{PrivEsc}) \Rightarrow \text{LateralMovement}$$

This matches the observed pattern in G_s, generating a symbolic inference.

5.8.3 Symbolic Inference Output

If a rule is satisfied, the symbolic engine generates an inference tuple:

$$O_i = (\text{AttackID}_i, \text{Conf}_i, \text{CausalPath}_i)$$

Example:

$$O_i = (\text{T1021}, 0.79, A \rightarrow B \rightarrow \text{DomainController})$$

The confidence score is computed using rule support and anomaly deviation:

$$\text{Conf}_i = \beta_1 \cdot \text{Score}_{\text{GNN}} + \beta_2 \cdot \text{Support}(r_i)$$

The causal path lists the sequence of events and nodes contributing to the decision, providing a structured trace that is later consumed by the LLM for explanation generation.

The symbolic reasoning process ensures that neural anomaly outputs are augmented with domain knowledge and causally grounded interpretations, forming a bridge between statistical learning and human-readable logic.

Algorithm 5 LLM-Based Explanation Generation and Validation

Input: Symbolic tuple $(AttackID, Confidence, Path)$
Output: Validated explanation y

Construct reasoning trace:

$$x = \{AttackID, Confidence, Path\}$$

Generate explanation: $y \leftarrow LLM(x)$
Validate explanation with symbolic semantics:
if $y \models AttackID$ **then**
    Accept y
else
    Reject and regenerate explanation
return y

5.9 LLM-Based Explainability Layer

The LLM-based explainability layer converts symbolic reasoning outputs into human-interpretable text. This stage operates after the symbolic engine has identified an attack pattern and produced a structured tuple that includes the inferred attack type, the confidence value, and the causal event chain. The objective of this layer is to provide analysts with clear and consistent explanations that preserve factual accuracy while remaining easy to understand. Unlike traditional post-hoc explanation methods that rely solely on learned neural representations, this layer grounds each textual summary in symbolic evidence, ensuring that model outputs remain traceable and verifiable. The following paragraphs detail the explanation generation process and the mechanism that validates the coherence of the LLM output.The generation and verification workflow is summarized in algorithm 5.

5.9.1 Explanation Generation

The LLM receives a reasoning trace:

$$x = \{r_i, V_s, E_s, t_s\}$$

Node A $\rightarrow$ Node B (SMB File Read)
Node B $\rightarrow$ Domain Controller (Kerberos Request)
Node A performs privilege escalation shortly after

Rule template:

$$(\text{FileRead} \wedge \text{KerberosReq} \wedge \text{PrivEsc}) \Rightarrow \text{LateralMovement}$$

where r_i is the matched rule, V_s and E_s represent the subgraph nodes and edges involved in the anomaly, and t_s contains timestamp information. The LLM uses this structured input to generate a text summary that describes the attack behaviour. Formally, the explanation generation process is represented by:

$$y = \text{LLM}(x)$$

The generated text follows a constrained instruction template that enforces factual grounding. For example, when rule r_{T1110} is activated, the LLM produces an output similar to:

> "The system detected repeated authentication failures followed by a successful login and outbound communication. These events correspond to the MITRE T1110 brute-force technique."

Example – Suspicious Lateral Movement
Suppose the anomalous subgraph is:

Symbolic Inference Output:

$$O = (\text{T1021}, 0.79, A \rightarrow B \rightarrow \text{DomainController})$$

The generation process preserves semantic consistency with the symbolic input while enhancing readability.

5.9.2 Symbolic Consistency Validation

To ensure that LLM outputs remain aligned with the detected evidence, each explanation is evaluated through a symbolic validation operator:

$$\text{Validate}(y, r_i) = \begin{cases} 1, & \text{if } y \models r_i \\ 0, & \text{otherwise} \end{cases}$$

The validation mechanism checks that all predicates referenced in the explanation appear in the symbolic trace and that no unsupported claims are introduced. Formally, the validation constraint requires:

$$\text{Predicates}(y) \subseteq \text{Predicates}(r_i)$$

and

$$\text{CausalEvents}(y) = \text{CausalPath}_i$$

If the explanation fails validation, the LLM is prompted to regenerate a revised summary until consistency is achieved. This ensures that textual outputs remain trustworthy and verifiable.

The LLM explainability layer therefore synthesises symbolic knowledge and natural-language generation in a controlled manner, producing clear and accurate descriptions of detected attacks while maintaining alignment with the underlying behavioural evidence.

5.10 Reinforcement Learning Controller

The reinforcement learning (RL) controller provides adaptive regulation of the system's operational parameters. Its purpose is to ensure that FedSymGraph maintains an optimal balance between accuracy, privacy, latency, and communication cost throughout the training process. Unlike fixed-parameter federated learning systems, FedSymGraph introduces a dynamic agent that observes system behaviour and updates its policy based on reward feedback. This allows the framework to adjust differential privacy noise, communication intervals, anomaly detection thresholds, and other parameters in response to environmental changes. The following paragraphs describe the state–action structure of the controller, its reward function, and the optimisation process. The optimisation process is described in Algorithm 6.

5.10.1 State and Action Space

At each training round t, the RL agent observes a state vector:

$$s_t = \{\text{Acc}_t, \text{FPR}_t, \text{TPR}_t, \text{Lat}_t, \text{Comm}_t, \sigma_t, \tau_t\}$$

where:

- Acc_t is detection accuracy
- FPR_t is the false positive rate
- TPR_t is the true positive rate
- Lat_t denotes round latency
- Comm_t refers to communication cost
- σ_t is the differential privacy noise scale
- τ_t is the anomaly detection threshold

Based on this state, the agent selects an action:

$$a_t = \{\Delta\sigma_t, \Delta\tau_t, \Delta C_t\}$$

where each Δ term represents an incremental increase or decrease in the corresponding parameter. The controller therefore regulates three essential behaviours: privacy strength, detection sensitivity, and communication frequency.

Example – High False Positives

If the false-positive rate increases:

Algorithm 6 Adaptive Control via Reinforcement Learning

Input: Performance metrics $(Acc_t, FPR_t, Comm_t)$
Output: Updated policy parameters $(\sigma, \tau, \Delta_{comm})$
1: Observe state:

$s_t = \{Acc_t,\ FPR_t,\ Comm_t,\ PrivacyBudget_t\}$

2: Compute reward:

$R_t = \alpha_1(1 - FPR_t) + \alpha_2(Acc_t) - \alpha_3(Comm_t)$

3: Update policy using PPO gradient ascent:

$\pi_{t+1} = \pi_t + \eta \nabla_\pi J(\pi_t)$

4: Extract new parameters $(\sigma, \tau, \Delta_{comm})$ from updated policy
5: **return** updated control parameters

$\mathrm{FPR}_t = 0.19$

reward decreases. The RL agent may reduce sensitivity by increasing threshold τ:

$a_t = \tau \leftarrow \tau + 0.04$

5.10.2 Reward Design

The reward function encourages high accuracy and low false positives while penalising communication overhead:

$$R_t = \alpha_1(1 - \mathrm{FPR}_t) + \alpha_2(\mathrm{TPR}_t) - \alpha_3(\mathrm{Comm}_t)$$

The coefficients $\alpha_1, \alpha_2, \alpha_3$ determine the trade-off between detection quality and communication efficiency. For example, a detection-focused deployment may choose $\alpha_2 > \alpha_1$, prioritising true positives over minimising false alarms.

To prevent privacy degradation, the RL agent must also satisfy the constraint:

$$\sigma_t \geq \sigma_{\min}$$

where $\sigma_{\min}$ ensures a minimum required privacy budget.

5.10.3 Policy Optimisation

The agent learns a policy $\pi_\theta(a_t \mid s_t)$ using Proximal Policy Optimisation (PPO). The objective maximises expected cumulative reward:

$$\max_\theta \mathbb{E}\left[\sum_{t=1}^{T} \gamma^t R_t\right]$$

PPO uses a clipped surrogate loss to ensure stable policy updates:

$$L_{\mathrm{PPO}}(\theta) = \mathbb{E}\Big[\min\left(r_t(\theta)A_t,\ \mathrm{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t\right)\Big]$$

where $r_t(\theta)$ is the probability ratio and A_t is the advantage estimate.

Example – PPO Update
If the advantage estimate is:

$A_t = +0.25$

and:

$r_t(\theta) = 1.12$

but clipping threshold is $1 + \epsilon = 1.10$, the clipped value is used:

$r_{t,\mathrm{clip}} = 1.10$

Through repeated optimisation, the controller converges to a policy that adapts the system parameters dynamically, improving both performance and resilience under evolving network conditions. This mechanism allows FedSymGraph to operate efficiently in settings with variable load, heterogeneous domains, and changing threat landscapes.

6 Security Analysis and Experiments

6.1 Security Analysis

This section evaluates the security guarantees provided by the FedSymGraph framework. The analysis considers confidentiality, integrity, transparency, robustness, fault tolerance, and scalability.

6.1.1 Privacy Preservation

FedSymGraph preserves privacy through Differential Privacy (DP), parameter masking, and encrypted communication channels. During local training, each domain perturbs its GNN-based model update using Gaussian noise:

$$\tilde{w}_k = w_k + \mathcal{N}(0, \sigma^2 I)$$

The privacy budget ε controls how much information can be inferred from transmitted updates. Only aggregated privacy-preserving updates are exchanged. Raw logs, temporal graphs, and anomaly traces remain local.

6.1.2 Secure Aggregation

Each domain submits a protected update $\tilde{w}_k$ to the aggregator. A weighted trust-aware averaging rule is applied:

$$w_{t+1} = \frac{\sum_{k=1}^{K} n_k q_k \tilde{w}_k}{\sum_{j=1}^{K} n_j q_j}$$

q_k is a reliability score derived from historical behaviour. Suspicious updates are filtered. All communication is encrypted. Model dissemination is protected through attribute-controlled access.

6.1.3 Transparency and Traceability

FedSymGraph records contributions, update statistics, and validation outcomes. Each update includes anomaly metadata identifying suspicious nodes, subgraphs, and temporal patterns. Digital signatures guarantee message integrity and authenticity. All events are logged for auditability and protocol compliance.

6.1.4 Resilience to Attacks

DP mitigates inference attacks. Secure aggregation prevents manipulation during transmission. Symbolic reasoning validates unusual graph structures before updates affect the global model. Contributions with abnormal behaviour are down-weighted through reduced q_k. The cross-layer design mitigates poisoning, evasion, and adversarial graph perturbations.

6.1.5 Fault Tolerance

The system supports redundant update collection and error-tolerant aggregation. Convergence does not require all domains in every round. Local temporal graph training continues during communication failures. The aggregator maintains versioned global models, allowing resumption without rollback.

6.1.6 Scalability with Security

Local GNN training scales linearly with the number of participants. Federated aggregation prevents central bottlenecks. DP and encrypted updates reduce communication. Trust-aware weighting filters unreliable domains. Symbolic reasoning adds explainability without significant overhead. The system supports large multi-domain deployments under strict security and privacy constraints.

6.2 Communication Cost Analysis

6.3 Experiments

7 Discussion and Future Work

8 Conclusion

Declarations

Ethics Approval Our study did not include any human and/ or animal studies. All datasets used in the paper are publicly available for research purposes.

Conflict of Interests The authors declare that they have no conflict of interest.

Data Availability All datasets used in this paper to support the findings are publicly available. Links are reported in the bibliography.

Funding This study was conducted without any financial support.

References