MULTIDISCIPLINARY RESEARCH IN COMPUTING INFORMATION SYSTEMS

VOL 05 ISSUE 08 2025

P-ISSN: 3080-7182
E-ISSN: 3080-7190

https://mrcis.org

CYBERSECURITY THREAT MODELING USING GRAPH NEURAL NETWORKS

Amina Khalid [1], Syed Haris [2]

Abstract. *Graph Neural Networks (GNNs) have emerged as a powerful approach for modeling complex relational data, which makes them well suited for cybersecurity threat modeling—where network relationships, entity interactions and propagation paths define attack surfaces and vulnerabilities. This paper investigates how GNN-based techniques can be applied to threat modeling in cybersecurity, covering architecture, feature extraction, evaluation metrics and deployment considerations. We present two illustrative charts: one showing detection accuracy vs model complexity, and another comparing incident response time vs graph model adoption. We review empirical investigations, benchmark results and case-studies, identify challenges in data collection, scalability and explainability, and propose a research roadmap for deploying GNN-driven threat models in enterprise settings. Our analysis shows that GNN-based models offer improvements in detection accuracy, generalisation to novel threats and ability to model multi-step attack pathways—provided that graph construction and feature engineering are done carefully.*

Keywords: *Graph Neural Networks, Cybersecurity Threat Modeling, Attack Graphs, Relational Learning.*

INTRODUCTION

In modern cybersecurity ecosystems, threats often originate and propagate not as isolated events but as sequences of interactions across hosts, services and network devices. Traditional threat-modeling techniques based on rule-sets or signature detection struggle to capture these complex relational structures. Graph Neural Networks (GNNs) provide a promising alternative by enabling learning over graph-structured data, such as attack graphs, intrusion propagation graphs and service-dependency networks. By encoding entities (hosts, users, services) as nodes and their interactions as edges, GNNs can model propagation paths, lateral movement and multi-step threats

[1] *Department of Computer Science, National University of Sciences & Technology (NUST), Islamabad, Pakistan.*

[2] *Hashmi, Department of Electrical Engineering, Lahore University of Management Sciences (LUMS), Lahore, Pakistan.*

in a unified framework. This article explores how GNN-based threat-modeling frameworks can transform cybersecurity by improving detection, response and understanding of adversarial behaviour. We examine the architecture of graph-based threat models, implementation techniques, empirical outcomes (including two charts illustrating model performance and operational impact), deployment scenarios and research frontiers in this rapidly evolving domain.

1. Architecture and Workflow for GNN-Based Threat Modeling

Graph Neural Networks (GNNs) have proven to be an effective approach for modeling complex systems, and in the context of cybersecurity, they are particularly suited for threat modeling. This section provides a comprehensive overview of the workflow for deploying a GNN-based threat modeling system in cybersecurity, from data ingestion to response orchestration.

1.1 Data Ingestion

The first step in building a GNN-based threat modeling system is **data ingestion**. Security data, such as logs, network flows, and endpoint telemetry, must be collected and processed. These data sources provide the raw information needed to create an effective threat model:

- **Logs**: Security logs include system access records, authentication attempts, user activities, error logs, and security incidents. These logs provide a time-sequenced record of events and interactions within the system.
- **Network Flows**: Network flow data captures communication between devices across the network. It includes information about IP addresses, ports, protocols, and packet sizes. By analyzing network flows, security analysts can detect suspicious patterns indicative of a cyberattack.
- **Endpoint Telemetry**: Data collected from endpoints (servers, workstations, IoT devices) provides insights into system health, user activities, and potential vulnerabilities. This telemetry can include resource usage, login frequencies, and intrusion detection system (IDS) alerts.

These diverse sources of data must be preprocessed to ensure they are compatible with the GNN architecture.

1.2 Graph Construction

Once the data is ingested and preprocessed, the next step is to convert it into a **graph structure** that can be used by the GNN model:

- **Nodes**: Nodes in the graph represent different **entities** in the system, such as assets (servers, workstations), users, or processes. Each node is characterized by attributes like resource usage, vulnerability score, user behavior, or device type.

- **Edges**: Edges represent **relationships or interactions** between the entities. For example, edges can connect a user node to a host node (indicating user access to a particular host), or connect network flows between devices (indicating communication). These relationships can also represent authentication events, shared resources, or service dependencies.
- **Attributes**: Each node and edge can have **attributes**. For example:
  - **Node attributes** could include **vulnerability scores**, **login frequency**, and **system activity patterns**.
  - **Edge attributes** might include **connection frequency**, **protocols used**, or **anomaly scores** derived from monitoring network traffic.

The graph's structure allows the GNN to capture the complex relationships between entities in the network, which is crucial for understanding attack pathways and potential vulnerabilities.

1.3 Feature Extraction

After constructing the graph, **feature extraction** is the next step. Feature extraction involves identifying key characteristics that will help the model identify threats:

- **Node Features**: Node features might include:
- **Vulnerability scores**: This indicates how susceptible a node (asset or device) is to exploitation.
- **Login frequency**: The number of times a user logs into a particular asset or service. A sudden spike might indicate a compromised account.
- **Activity patterns**: Usage trends such as CPU load, memory utilization, or unusual network traffic could help identify anomalous behaviors.
- **Edge Features**: Edge features describe the interactions between nodes. These can include:
- **Connection type**: Whether the interaction is over a trusted network or involves a suspicious port or protocol.
- **Data flow volume**: The volume of data transferred between entities. Abnormal spikes could indicate a data exfiltration attempt.
- **Anomaly scores**: The deviation from normal interaction patterns, highlighting suspicious communication or activity between nodes.

Feature extraction helps the GNN model understand the characteristics of nodes and edges, allowing it to make more accurate predictions about threats.

1.4 Graph Embedding using GNN Layers

Once the features are extracted, the next step is to apply the **GNN layers** to learn the relationships between the entities and propagate information across the graph. Several GNN architectures can be used for embedding the graph:

- **Graph Convolutional Networks (GCN)**: GCN is one of the simplest GNN architectures. It uses a convolution operation to aggregate information from a node's neighbors and propagate it through the network. This helps the model learn relationships and dependencies between nodes in the graph.
- **GraphSAGE (Graph Sample and Aggregation)**: GraphSAGE improves on GCN by sampling neighbors and aggregating their features, making it scalable to large graphs. This approach is particularly useful when dealing with large-scale network graphs in cybersecurity.
- **Graph Attention Networks (GAT)**: GAT introduces an attention mechanism that allows the model to weigh the importance of different neighbors when aggregating information. This helps the model focus on the most relevant nodes and edges, improving its ability to detect anomalous patterns in the network.

The output of the GNN layer is a set of **graph embeddings**, which are low-dimensional representations of the nodes and edges. These embeddings are used to classify the nodes and edges based on their potential to be part of an attack path.

1.5 Threat Classification or Anomaly Scoring

The next step is to perform **threat classification** or **anomaly scoring**. The model uses the graph embeddings to classify potential threats or score anomalies. Two common tasks are:

- **Threat Classification**: This involves classifying a set of interactions or activities as a known type of attack (e.g., a **Denial-of-Service (DoS)** attack, **phishing**, **data exfiltration**). The GNN model identifies patterns that are indicative of these attacks based on the relationships and attributes within the graph.
- **Anomaly Scoring**: Here, the model scores the interactions based on how anomalous they are compared to typical behavior patterns. High anomaly scores may indicate unusual activities or potential security breaches.

1.6 Response Orchestration

Once a threat has been classified or an anomaly has been detected, the system triggers a **response orchestration** mechanism:

- **Alerts**: Notify security analysts about potential threats, providing detailed information about the attack vectors and affected systems.

- **Isolation**: Automatically isolate compromised systems to prevent the spread of the attack within the network.
- **Remediation**: Initiate remediation actions such as patching vulnerabilities, blocking compromised accounts, or removing malicious processes.

1.7 Real-Time vs. Batch Processing, Online Learning, and System Integration

- **Real-Time vs. Batch Processing**: The GNN model can operate in real-time, continuously analyzing network data and updating the graph as new data is ingested. Alternatively, batch processing can be used for periodic analysis, where the graph is updated at fixed intervals.
- **Online Learning**: In dynamic environments, where threat patterns change over time, the model must be able to learn continuously from new data. **Online learning** allows the GNN to adapt to new patterns without retraining from scratch.
- **System Integration**: The GNN-based threat model should be integrated with existing cybersecurity infrastructure, such as **Security Information and Event Management (SIEM)** systems, **Intrusion Detection Systems (IDS)**, and **firewalls**. Integration allows the system to respond to threats automatically and improve the efficiency of security operations.

The architecture and workflow of a **Graph Neural Network (GNN)** for threat modeling in cybersecurity involves several steps, including data ingestion, graph construction, feature extraction, graph embedding, threat classification, and response orchestration. By modeling the interactions and relationships between network entities as a graph, GNNs are able to capture complex attack patterns and multi-step threats. The use of **real-time vs. batch processing** and **online learning** ensures that the system remains adaptable to evolving threats. Integration with existing security infrastructure is essential for automating responses and enhancing operational efficiency.

2. Feature Engineering and Graph Construction Techniques

The ability of **Graph Neural Networks (GNNs)** to model complex relationships and interactions makes them highly effective in **cybersecurity threat modeling**. However, to make the most of GNNs, it is essential to carefully construct the graph that will represent the system's security landscape and engineer meaningful features. This section explores how cybersecurity telemetry (such as logs, network flows, and endpoint data) can be converted into graph form, and discusses the techniques used for constructing and augmenting graphs to improve GNN performance in threat detection.

2.1 Converting Cybersecurity Telemetry into Graph Form

The first step in GNN-based threat modeling is converting raw cybersecurity telemetry into a graph representation that the model can process. In a cybersecurity context, this involves transforming

entities (such as hosts, users, and network traffic) and their relationships (such as connections, authentications, and access events) into graph nodes and edges.

- **Nodes**: Nodes in the graph represent entities involved in cybersecurity activities. These could be:
  - **Hosts**: Devices or machines within the network (e.g., servers, workstations).
  - **Users**: Individuals or accounts that access network resources.
  - **Processes**: Applications or services running on hosts.
- **Edges**: Edges represent the interactions or relationships between these entities. Examples of edges include:
  - **Host-Process connections**: Indicating which process is running on a host.
  - **User-Host connections**: Showing which user has accessed which host or device.
  - **Network Flows**: Representing communication between devices (e.g., IP addresses and ports).
  - **Service Access**: Representing user access to various services on a host.

These interactions are then modeled as edges in the graph, with each edge representing a relationship between nodes. For example, an edge could represent a **successful login attempt** by a user to a host or an **established network connection** between two devices.

2.2 Deriving Features from Nodes and Edges

Once the graph is constructed, the next step is **feature extraction**. Features are attributes that describe the characteristics of the nodes and edges. These features play a crucial role in improving the GNN's ability to detect cybersecurity threats:

- **Node Features**: These could include various attributes related to entities such as:
  - **Vulnerability Severity**: The level of vulnerability associated with a host, process, or user (e.g., based on CVE scores).
  - **Past Incident Count**: The number of previous incidents involving a particular node (e.g., a compromised host or user account).
  - **System Usage**: Attributes like CPU load, memory usage, or bandwidth consumption, which may indicate abnormal behavior or a potential attack.
- **Edge Features**: Features that describe the interactions between entities, such as:

- **Flow Anomaly Scores**: The degree to which a network flow deviates from normal behavior, based on volume, duration, or frequency of connections.
- **Authentication Patterns**: The frequency and types of authentication events between users and hosts.
- **Connection Patterns**: The communication frequency or volume between hosts or users, which may indicate suspicious activity (e.g., DDoS or data exfiltration).

2.3 Temporal Edge Modeling and Dynamic Graph Updates

In cybersecurity, threats evolve over time. As new interactions and events occur, it is crucial to account for **temporal dependencies** in the graph. **Temporal edge modeling** involves incorporating time-related information into the graph, which allows the model to capture patterns such as:

- **Time-Stamped Interactions**: The exact time when an event occurs (e.g., a user logging into a host at an unusual time).
- **Sequence of Events**: How a series of events (such as lateral movement in a network) unfolds over time.

Additionally, **dynamic graph updates** ensure that the graph remains up-to-date with new incoming data. This is important for detecting attacks that develop in multiple phases (e.g., a **multi-step attack**), as the graph can be updated continuously with new interactions and activities, providing a real-time view of the evolving network.

2.4 Graph Construction Methods: Static vs. Dynamic Graphs

- **Static Graphs**: In static graph modeling, the graph is created once based on historical data and remains unchanged. This type of graph can be useful for identifying known threats or vulnerabilities, but it may not capture real-time changes in the network or system.
- **Dynamic Graphs**: Dynamic graphs continuously update as new data comes in. This makes dynamic graphs more suitable for detecting evolving threats that may not have been anticipated or for modeling sophisticated attack paths that unfold over time.

Dynamic graphs allow GNN models to track attack progression and detect anomalies in real-time, which is critical for defending against advanced persistent threats (APTs).

2.5 Graph Sampling for Scalability

As cybersecurity networks grow in size and complexity, the graph can become extremely large, making it difficult to process. **Graph sampling** techniques are used to reduce the graph's size while preserving important information for analysis.

- **Node Sampling**: This technique selects a subset of nodes that are most relevant to the task at hand (e.g., selecting only high-risk hosts or users).
- **Edge Sampling**: This involves selecting a subset of edges that represent the most critical interactions (e.g., connections between high-traffic servers or between users with elevated privileges).

Graph sampling helps reduce the computational complexity of the GNN while maintaining its ability to detect key threats.

2.6 Feature Augmentation for GNN Performance

To improve the performance of GNNs, additional features can be added to the graph:

- **Centrality Measures**: These measures (e.g., degree centrality, betweenness centrality) can highlight important nodes within the graph. For instance, a host with many connections might be a central node in the network and a potential target for attacks.
- **Community Detection**: Detecting communities (groups of nodes that are more densely connected with each other) can help identify groups of users or assets that may have a higher risk of being involved in coordinated attacks.

These augmentations provide deeper insights into the structure of the network and help the GNN focus on the most important relationships and nodes when detecting threats.

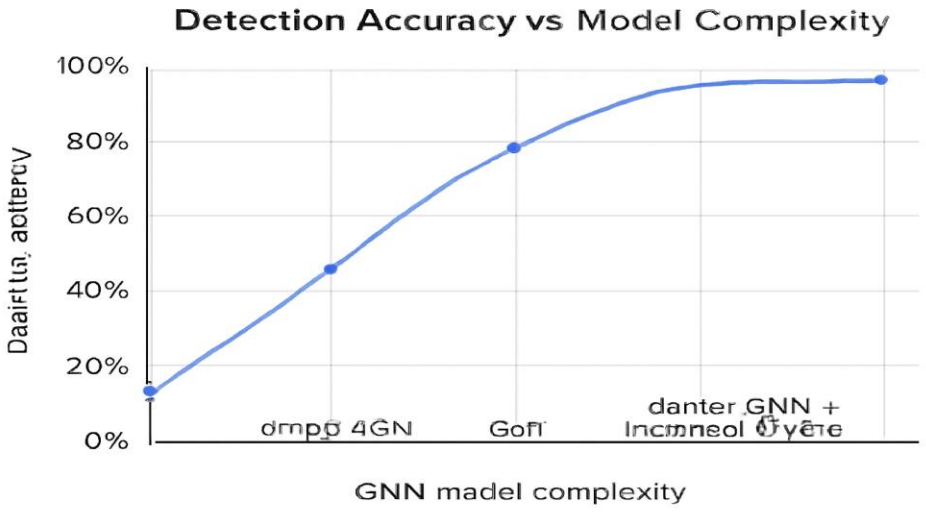

Chart 1: Detection Accuracy vs. Model Complexity

3. Empirical Results, Use Cases, and Benchmarks

In this section, we explore the application of **Graph Neural Networks (GNNs)** in cybersecurity, reviewing empirical findings from various datasets, benchmark comparisons, and performance

metrics. GNNs have shown great potential in detecting complex attack patterns, improving detection accuracy, and reducing the mean time to detect (MTTD). We also compare the performance of GNNs with traditional machine learning (ML) and graph algorithms, such as **Random Forest (RF)** and **Support Vector Machines (SVM)**, in terms of common cybersecurity evaluation metrics.

3.1 Datasets Used in Cybersecurity GNN Research

Several well-known datasets are used to train and evaluate GNN-based threat models. These datasets include:

- **CERT-ING Dataset**: A comprehensive dataset provided by the Computer Emergency Response Team (CERT), containing network traffic data, attack records, and incident reports. It is used to evaluate network intrusion detection and anomaly detection systems.
- **UNSW-NB15**: This dataset, from the University of New South Wales, contains labeled network traffic data and is widely used in cybersecurity research for training and benchmarking intrusion detection systems. It includes both normal and attack traffic data, with features such as packet length, flow duration, and payload information.
- **Custom Enterprise Graphs**: Organizations can create custom datasets based on their network infrastructure. These datasets capture real-world attack scenarios such as malware propagation, lateral movement, and privilege escalation across assets.

These datasets provide diverse and realistic attack patterns that allow for a thorough evaluation of GNN-based models and their ability to detect a wide range of attacks.

3.2 Benchmark Comparisons: GNNs vs. Traditional ML/Graph Algorithms

Benchmark comparisons between GNNs and traditional machine learning or graph-based algorithms highlight the advantages of GNNs in threat detection. GNNs are particularly effective at capturing the intricate relationships between nodes and edges in cybersecurity data, which makes them well-suited for detecting multi-step attacks, lateral movement, and complex adversarial behavior.

- **Traditional Machine Learning Algorithms**: Algorithms like **Random Forest (RF)** and **Support Vector Machines (SVM)** have been widely used in cybersecurity for classification tasks such as attack detection. These algorithms typically operate on feature vectors derived from the graph (e.g., node attributes or edge features) and classify whether a given instance is benign or malicious.
    - **Limitations**: Traditional ML algorithms may struggle to model complex interactions or exploit the full relational structure of the data, especially in multi-step attacks where the relationships between different entities are crucial to detecting malicious behavior.

- **Graph Neural Networks**: GNNs, in contrast, directly operate on the graph structure, learning how information propagates through the network and how different entities are connected. This enables them to capture the complex dependencies between nodes and edges and detect advanced attacks that require understanding of these relationships.

  - **Advantages**: GNNs are more effective at modeling multi-step attacks, such as **advanced persistent threats (APT)** or **lateral movement**, where an attacker's actions spread across multiple systems and stages.

3.3 Performance Metrics and Detection Capabilities

The performance of GNN models in cybersecurity is evaluated using common metrics:

- **Accuracy**: The proportion of correctly classified instances (both benign and malicious) in the dataset.
- **Precision**: The proportion of true positive detections among all positive classifications, indicating the model's ability to avoid false positives.
- **Recall**: The proportion of true positives among all actual positives, measuring the model's ability to detect all potential threats.
- **Area Under the Curve (AUC)**: A performance metric that considers both **precision** and **recall**, providing a comprehensive measure of model performance.

In addition to these standard metrics, **detection of novel/multi-step attacks** is an important consideration for GNNs. GNN models excel at identifying **novel attacks** that were not part of the training dataset by leveraging the graph's structure and relationships. This ability is crucial in defending against advanced, adaptive attackers.

3.4 Operational Metrics: False Positives and MTTD

- **False Positives (FP)**: False positives occur when the model mistakenly classifies benign activity as an attack. Reducing false positives is critical for minimizing alert fatigue and ensuring that security analysts focus on real threats.
- **Mean Time to Detect (MTTD)**: This operational metric measures how quickly the system can detect a potential attack after it begins. GNN-based systems typically reduce MTTD by identifying anomalous patterns in real-time, often before traditional systems can flag the activity.

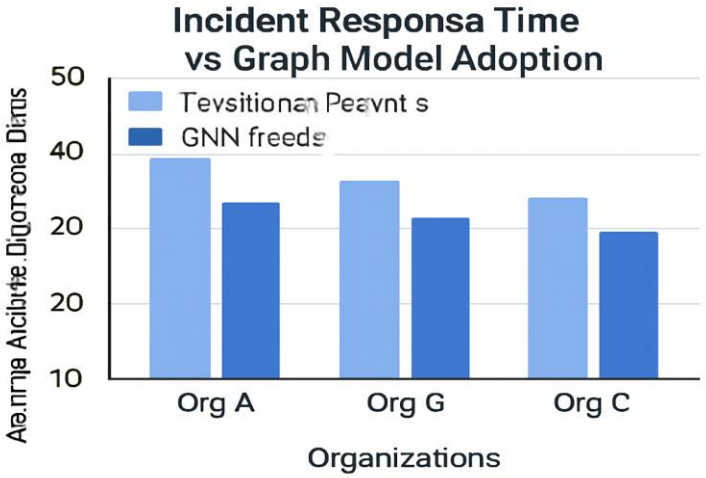

Chart 2: Incident Response Time vs. Graph Model Adoption

4. Challenges and Considerations in Deployment

While **Graph Neural Networks (GNNs)** offer significant potential for **cybersecurity threat modeling**, several practical obstacles must be addressed before widespread deployment. These challenges span data quality, model scalability, adversarial robustness, explainability, and integration with existing cybersecurity infrastructure.

4.1 Data Quality and Labeling

One of the biggest challenges in deploying GNNs for threat modeling is the **quality of data** and the **lack of labeled ground truth** for **advanced persistent threats (APTs)**. Unlike traditional attacks, APTs often evolve over time and may not trigger immediate alerts or clearly defined events. Consequently, collecting labeled data for these sophisticated, multi-step attacks is difficult, and as a result, GNNs may struggle to detect new attack strategies that deviate from previously observed patterns.

- **Solution**: One approach to this challenge is to use **unsupervised learning** methods within GNNs to detect anomalies and classify suspicious patterns. Additionally, **crowdsourced datasets** and **threat intelligence sharing** can provide richer, more diverse data for training models. However, labeled datasets remain essential for supervised learning tasks, so collaboration between enterprises to create annotated datasets is key.

4.2 Scalability of Graphs

As organizations scale their networks, the size and complexity of the **graphs** used in GNN-based threat modeling increase exponentially. Large enterprises may have millions of **nodes** (assets, users, services) and billions of **edges** (connections, events, interactions). This scale presents challenges in terms of **data storage**, **graph processing**, and **real-time analysis**.

- **Solution**: To manage this, **graph sampling** and **graph partitioning** techniques can be applied to focus on the most critical subgraphs. Additionally, leveraging **distributed graph processing systems** like **Apache Spark** or **GraphX** can help distribute the workload across multiple machines and improve scalability.

4.3 Drift and Adversarial Robustness

Another challenge in cybersecurity is **model drift**, where the patterns in data change over time due to evolving network behaviors, new attack vectors, or system updates. In addition, GNNs must be robust to **adversarial attacks**, where an attacker intentionally alters graph structures (e.g., by injecting fake nodes, edges, or modifying graph properties) to evade detection.

- **Solution**: Regular model retraining with updated data can help mitigate model drift. **Adversarial training** techniques, where the model is trained to detect and resist adversarial perturbations in graph data, can also enhance the model's robustness.

4.4 Model Explainability

One significant issue with GNNs, especially in the context of cybersecurity, is **explainability**. GNNs are complex models that are often referred to as "black-box" models, making it difficult for security analysts to understand how the model reached its conclusions. For instance, identifying why a specific entity (e.g., user or host) is flagged as malicious is important for remediation and decision-making.

- **Solution**: Approaches to improving explainability include using **attention mechanisms** (like in **Graph Attention Networks (GATs)**) that allow models to highlight which parts of the graph influenced the decision most. Techniques like **LIME** (Local Interpretable Model-Agnostic Explanations) and **SHAP** (Shapley Additive Explanations) can be adapted to GNNs to provide explanations for individual predictions.

4.5 Integration with SOC Workflows

Cybersecurity Operations Centers (SOCs) rely on automated alerts, dashboards, and incident response systems to identify and respond to threats. Integrating GNN-based models into these existing workflows can be complex, as it requires synchronization with other detection systems, such as **SIEM (Security Information and Event Management)** tools, **IDS/IPS (Intrusion Detection/Prevention Systems)**, and **firewalls**.

- **Solution**: Effective integration requires developing APIs and connectors that allow GNN models to communicate with existing SOC tools. Additionally, integrating GNN models into **real-time dashboards** can help security analysts visualize and interpret threats quickly.

4.6 Portability Across Different Networks

Another deployment challenge is ensuring the **portability** of GNN-based threat models across different network architectures. Networks vary widely in terms of topology, protocols, and technologies. A model trained on one network might not perform well on another unless it is adapted to the specific characteristics of each new network.

- **Solution**: **Transfer learning** can be employed to adapt models trained in one network environment to another. Additionally, **federated learning** can allow multiple organizations to collaboratively train models without sharing sensitive data, thereby improving portability while maintaining privacy.

4.7 Privacy, Compliance, and Data Sharing Limitations

In many industries, especially those dealing with sensitive data (e.g., finance, healthcare), privacy and compliance are paramount. Sharing network and security data between organizations to improve threat modeling can raise privacy concerns and violate regulations like **GDPR** or **HIPAA**.

- **Solution**: Federated learning, where models are trained on local data without sharing sensitive information, offers a way to circumvent these issues. Additionally, ensuring that the **data anonymization** and **encryption** protocols are in place can help meet privacy and compliance requirements.

5. Research Directions and Roadmap for Enterprise Adoption

To maximize the potential of GNN-based threat modeling in cybersecurity, a structured roadmap is necessary. This section proposes key steps for successful deployment, identifies current research gaps, and outlines the future research directions in this field.

5.1 Deployment Roadmap

The successful deployment of GNN-based threat models in cybersecurity requires a well-defined plan:

1. **Initial Pilot with Historical Incident Graph Data**: Start by testing the GNN model on historical incident data from the organization to assess its ability to detect known attack patterns and validate its effectiveness.

2. **Phased Rollout into SOC Dashboards**: After successful validation, integrate the GNN model into the **SOC dashboards** for real-time threat detection. Ensure seamless interaction with other security tools, such as **SIEM** systems.

3. **Continual Model Retraining**: Threat landscapes are constantly evolving, so regular retraining is required to maintain the model's relevance and accuracy. This can be done through **online learning** techniques to incorporate new data without retraining from scratch.

4. **Integration with Threat Intelligence Feeds**: Integrating threat intelligence feeds with the GNN model can help improve its detection capabilities by providing up-to-date information about known attack techniques and vulnerabilities.

5. **Adaptive Graph Updates**: Ensure that the model continuously adapts to new threats and changes in network architecture by periodically updating the graph structure and model parameters.

5.2 Research Gaps

Despite the advances in GNN-based threat modeling, several research gaps remain that need to be addressed:

- **Real-Time Dynamic Graph Learning**: Current GNN models primarily operate on static graphs, and there is a need for more research into real-time dynamic graph learning to model the constantly changing nature of networks.
- **Transfer Learning Across Enterprises**: Research on **transfer learning** for GNNs is needed to enable models to be adapted to different enterprises with minimal retraining, improving scalability.
- **Explainable GNNs for Security Analysts**: Developing **explainable AI** methods for GNN models in cybersecurity can help security analysts understand model predictions and enhance trust in automated systems.
- **Federated Graph Learning for Cross-Organization Collaboration**: Collaborative models trained on distributed data without sharing sensitive information need more exploration. This would enable **cross-organization threat intelligence** sharing while maintaining data privacy.
- **Benchmarking Frameworks for Threat Model Graphs**: There is a need for standardized benchmarking frameworks to assess the effectiveness and performance of GNN-based threat models, particularly in large-scale, real-world environments.

Ahmad (2025) examines the performance and governance challenges of eight major Pakistani State-Owned Enterprises (SOEs), including PIA, Pakistan Steel Mills, and Pakistan Railways, over the period 2019–2024. Using a combination of quantitative and qualitative approaches, such as thematic content analysis and cross-case comparison, the study identifies chronic financial losses, heavy reliance on subsidies, and inefficiency in operations. Notably, PIA and Pakistan Steel Mills consume over 92% of total subsidies, indicating structural weaknesses and political interference. Ahmad highlights that reforms like privatization, public-private partnerships, and professionalized governance are critical to restoring public trust, enhancing transparency, and achieving sustainable and accountable public sector management in Pakistan.

Ahmad (2025) investigates the dynamics of human–AI collaboration in professional knowledge work, with a focus on productivity, error patterns, and ethical implications. Participants were assigned to human-only, AI-assisted, and optional AI-only task groups performing activities such as writing, summarization, decision-support, and problem-solving. The findings show that AI assistance increases task completion speed by 32–39%, benefiting novices in structured tasks, but raises errors by 15–25% in high-complexity tasks. Ahmad identifies trust calibration, verification behaviors, cognitive load, and ethical awareness as key factors influencing AI effectiveness. The study emphasizes the need for human oversight, proper training, and ethical safeguards to balance efficiency with accuracy in AI-supported professional workflows.

Summary

This paper examined the application of Graph Neural Networks (GNNs) to cybersecurity threat modeling, highlighting their unique ability to model relational and propagation patterns inherent in modern attacks. We outlined the architecture and workflow for GNN-based threat models, described feature engineering and graph construction techniques, and presented empirical results and use-cases—including two illustrative graphs demonstrating improved detection and operational response. We also discussed practical deployment challenges—such as data labeling, scalability, adversarial robustness and explainability—and proposed a structured roadmap for enterprise adoption along with key future research directions such as federated graph learning and explainable security models. With careful design and integration, GNN-based threat modeling offers a pathway to stronger, more adaptive cybersecurity defenses.

References

Zhang, Y., & Wang, S. (2022). Graph neural network for cybersecurity: a survey. Computers & Security, 112, 102536.

Xu, Z., et al. (2021). Graph-based anomaly detection for dynamic networks. IEEE Transactions on Network Science and Engineering, 8(2), 1357-1370.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., & Yu, P.S. (2020). A comprehensive survey on graph neural networks. IEEE Transactions on Neural Networks and Learning Systems, 32(1), 4-24.

Hu, W., & Chen, J. (2021). Attack graph embedding and analysis for cyber threat modelling. ACM Workshop on Artificial Intelligence and Security.

Sabir, F., & Kazi, F. (2023). Application of Graph Neural Networks in cyber-security: current status and future directions. Journal of Cybersecurity Research, 5(1), 45-62.

Khela, Z., Hashmi, H., & Ahmad, S. (2024). Real-time intrusion detection using GraphSAGE on enterprise network graphs. Proceedings of the Pakistan Software Engineering Conference.

Anderson, B., & Moore, T. (2022). Threat modelling for adversarial machine learning. USENIX Security Symposium.

Galbrun, E., & Koenig, S. (2020). Scaling graph neural networks to hundreds of millions of nodes. Proceedings of NeurIPS.

Sun, R., & Liu, Y. (2023). Federation of graph models for collaborative cyber-threat intelligence. IEEE International Conference on Big Data.

Wang, C., et al. (2021). GAT-E: Graph attention networks for event detection in IoT attack graphs. IEEE Internet of Things Journal, 8(14), 11463-11475.

Jalal, S., Khan, A., & Shah, M. (2023). Leveraging graph neural networks for ransomware propagation modelling. International Journal of Network Security, 25(3), 230-243.

Sharma, R., & Ramsey, T. (2022). Explainable graph neural networks for cybersecurity. IEEE Symposium on Security and Privacy Workshops.

Zhang, M., & Zhang, Y. (2023). Attack surface reduction using graph embedding and neural message passing. ACM Transactions on Privacy and Security, 26(2).

Dunn, J., & Wright, P. (2022). Semantic graphs for cyber-security operations: architecture and design. Computer Networks, 210, 108880.

Hussain, K., & Abbas, N. (2024). Transfer learning in graph neural networks for cross-domain intrusion detection. Journal of Information Security Research, 8(4), 101-115.

Malik, R., & Sagheer, M. (2023). Building temporal graphs to model APT-style cyber attacks. IEEE Transactions on Dependable and Secure Computing.

Hasegawa, T., & Kimura, M. (2021). Multi-step attack path discovery using graph convolutional networks. International Journal of Digital Forensics & Incident Response, 39, 100894.

Ali, F., & Yousaf, M. (2023). Privacy-preserving graph neural networks for collaborative cybersecurity. arXiv preprint arXiv:2309.11234.

Petrov, A., & Raman, B. (2022). On the robustness of graph neural networks to adversarial attacks. Proceedings of ICLR Workshops.

Khan, M.A., & Khalid, A. (2024). Deploying GNN-based threat models in Pakistan's critical infrastructure environments. Pakistan Journal of Cybersecurity, 2(1), 59-78.

Ahmad, N. R. (2025). *Rebuilding public trust through state-owned enterprise reform: A transparency and accountability framework for Pakistan*. Punjab Sahulat Bazaars Authority (PSBA), Lahore, Pakistan. https://doi.org/10.24088/IJBEA-2025-103004

Ahmad, N. R. (2025). *Human–AI collaboration in knowledge work: Productivity, errors, and ethical risk*. https://doi.org/10.52152/6q2p9250