

Graph-Based Anomaly Intelligence for Detecting Coordinated Cyber Attacks in Distributed Smart Computing Environments

Dr. Tian Li

Assistant Professor of Computer Science & Data Science, University of Chicago, USA

Email: litian@uchicago.edu

Article Received 11-04-2026, Revised 09-05-2026, Accepted 03-06-2026

Author(s) Retains the Copyrights of This Article

Abstract

The rapid integration of cloud platforms, edge nodes, Internet of Things devices, industrial controllers, intelligent gateways, autonomous systems, and enterprise services has created distributed computing environments in which security events are increasingly interdependent. Conventional intrusion detection systems frequently process network flows, authentication events, process activities, and device alerts as individual observations. Such an approach can overlook coordinated attacks in which each individual event appears relatively benign while the collective sequence reveals lateral movement, privilege escalation, command-and-control activity, or multi-stage compromise. This study proposes a graph-based anomaly intelligence framework for identifying coordinated cyber attacks in distributed smart computing environments. The proposed framework represents devices, users, services, processes, communication endpoints, and security events as interconnected graph entities, while temporal, behavioral, and contextual information is associated with graph relationships. A dynamic graph construction mechanism continuously updates the security representation as communication patterns and system states evolve. Graph-based representation learning is subsequently employed to capture structural dependencies and behavioral deviations that may not be visible through independent event analysis. The framework combines node-level, edge-level, and subgraph-level anomaly evidence to generate an integrated anomaly intelligence score. Temporally related suspicious entities are then correlated to construct candidate attack paths and coordinated attack groups. The resulting approach is intended to distinguish isolated abnormalities from multi-entity attack campaigns. The framework accommodates heterogeneous security telemetry and changing network topologies while reducing dependence on manually defined signatures. Particular emphasis is placed on temporal consistency, false-positive control, scalability, explainability, and attack-path reconstruction. The proposed architecture is applicable to smart IoT infrastructures, edge computing environments, industrial networks, intelligent transportation systems, cloud platforms, and distributed enterprise systems. The study establishes a methodological foundation for adaptive cyber-defense systems that use structural and behavioral relationships to identify coordinated and multi-stage threats.

Keywords: Graph-Based Anomaly Detection, Graph Neural Networks, Coordinated Cyber Attacks, Cybersecurity Intelligence, Dynamic Graphs, Intrusion Detection, Distributed Computing, Smart Computing Environments, Attack Correlation, Heterogeneous Networks, Cyber Threat Detection, Graph Representation Learning.

1. Introduction

The expansion of distributed computing has fundamentally changed the structure of modern information infrastructures. Applications that were previously executed within relatively centralized data centers are now distributed across cloud services, edge computing platforms, Internet of Things devices, intelligent gateways, industrial controllers, mobile endpoints, and autonomous systems. These components exchange information continuously, creating highly interconnected environments in which a security incident affecting one entity can influence

the behavior of several other entities. Consequently, cybersecurity monitoring can no longer be viewed exclusively as the examination of isolated network packets, system calls, authentication records, or endpoint alerts. The relationships among entities and the evolution of those relationships over time have become important sources of security information. Graph-based learning provides a natural representation for this type of environment because a graph can express entities as nodes and their interactions as edges. Kipf and Welling demonstrated that graph convolutional networks can learn

representations directly from graph structure and node attributes, establishing an important foundation for graph-based representation learning [1]. Veličković et al. subsequently introduced graph attention networks, enabling learned attention weights to distinguish the relative importance of neighboring nodes [2]. Hamilton, Ying, and Leskovec proposed GraphSAGE as an inductive representation-learning approach capable of generating embeddings for previously unseen nodes, an important property for continuously changing distributed environments [3]. These developments provide useful mechanisms for transforming relational security telemetry into representations that can subsequently support anomaly detection and classification.

Anomaly detection in graphs presents a different problem from conventional point-based anomaly detection because abnormality may arise from an unusual node, an unusual relationship, an unusual local structure, or a combination of otherwise normal interactions. Deep Graph Infomax demonstrated that self-supervised graph representation learning can obtain informative representations without requiring complete manual labeling [4]. Zhao et al. extended this direction through Deep Graph Structural Infomax, incorporating structural information into representation learning and emphasizing the importance of local and edge-level structural characteristics [5]. Such approaches are relevant to cybersecurity because security datasets are often incomplete, imbalanced, heterogeneous, and expensive to label.

Earlier graph anomaly research also demonstrated the importance of structural patterns. Akoglu, McGlohon, and Faloutsos proposed OddBall for detecting unusual structural characteristics in weighted graphs [6]. Perozzi, Al-Rfou, and Skiena introduced DeepWalk, demonstrating that graph topology could be transformed into useful distributed representations [7]. Eswaran et al. developed SpotLight for anomaly detection in streaming graphs, addressing settings in which graph structure evolves continuously [8]. Manzoor, Milajerdi, and Akoglu proposed StreamSpot for streaming heterogeneous graphs and specifically considered the problem of identifying anomalous graph behavior in security-related environments [9]. These studies collectively indicate that graph structure can provide information that is difficult to capture when events are analyzed independently. StreamSpot, for example, was designed for heterogeneous graph streams and emphasized online anomaly scoring under bounded memory constraints.

The cybersecurity domain presents additional challenges because distributed attacks frequently involve multiple entities and stages. An attacker may initially compromise one device, establish persistence, obtain credentials, communicate with another endpoint, move laterally, and subsequently interact with a service or data repository. Each activity may be individually ambiguous, whereas their temporal and structural relationship can provide stronger evidence of coordinated behavior. Graph-based intrusion detection therefore offers an opportunity to combine communication topology, behavioral attributes, temporal information, and contextual relationships within a unified analytical representation.

Recent research has demonstrated the applicability of graph neural networks to network intrusion detection. Lo et al. proposed E-GraphSAGE, which incorporates both graph topology and edge features for intrusion detection in IoT environments [12]. Their work specifically represents network-flow information as graph data and evaluates the approach on multiple NIDS datasets. Ding et al. investigated inductive anomaly detection on attributed networks and developed an approach designed to generalize to newly observed nodes [10]. Ahanger et al. subsequently examined graph attention networks for IoT intrusion detection, showing the relevance of graph-based learning to heterogeneous IoT traffic [15].

Despite these developments, several research challenges remain. First, many approaches concentrate primarily on classification of individual network flows rather than coordinated attack campaigns. Second, graph construction itself remains a significant design problem because the same security environment can be represented using devices, IP addresses, users, processes, sessions, services, or combinations of these entities. Third, distributed environments are dynamic, meaning that static graphs may become outdated as devices join or leave, communication relationships change, and legitimate workloads evolve. Fourth, a high anomaly score alone does not necessarily explain why a particular group of entities is suspicious. Security analysts require interpretable evidence, including the relationships and temporal sequence that resulted in an alert.

This study addresses these challenges by proposing a graph-based anomaly intelligence framework that combines dynamic graph construction, graph-based behavioral representation learning, multi-level anomaly scoring, temporal correlation, and coordinated attack identification. Rather than treating an alert as an isolated classification event, the

proposed framework attempts to establish relationships among suspicious nodes, edges, and subgraphs and subsequently reconstruct potential attack paths.

The main objective is therefore to develop a unified methodology capable of identifying coordinated malicious behavior within distributed smart computing environments. The framework emphasizes five requirements: representation of heterogeneous security entities, continuous graph updates, multi-level anomaly analysis, temporal correlation of suspicious behavior, and interpretable attack-path analysis. The proposed design is intended as a general framework rather than a solution restricted to one network architecture or one attack family.

2. Literature Survey

Kipf and Welling introduced graph convolutional networks as a method for learning representations directly from graph structure and node attributes. Their formulation demonstrated that graph convolution could propagate information across neighboring nodes while maintaining computational scalability with respect to the number of graph edges. For cybersecurity applications, this principle is valuable because security entities rarely operate independently. A compromised endpoint, for example, may be recognized partly through the behavior of its communication neighbors. GCN-based propagation therefore provides a foundation for combining local structural and attribute information within a security graph [1].

Veličković et al. developed Graph Attention Networks to allow graph-learning models to assign different weights to neighboring entities during representation learning. The attention mechanism is particularly relevant to security environments in which some relationships are considerably more informative than others. A device communicating with an ordinary service, for example, may provide less anomaly evidence than the same device establishing an unusual connection with a previously unseen administrative endpoint. Attention-based learning consequently provides a mechanism for emphasizing relationships according to their learned relevance rather than treating all neighbors uniformly [2].

Hamilton, Ying, and Leskovec proposed GraphSAGE as an inductive framework in which node representations are generated from features aggregated from local neighborhoods. Unlike approaches that learn a fixed embedding for every node present during training, GraphSAGE learns an aggregation function that can be applied to previously unseen nodes. This

property is particularly useful for distributed computing environments because new devices, services, containers, users, and endpoints may continuously appear in the monitored infrastructure [3].

Veličković et al. proposed Deep Graph Infomax, which demonstrated the potential of self-supervised graph representation learning for obtaining informative node embeddings without relying exclusively on manually labeled data. The use of self-supervised learning is relevant to cybersecurity because obtaining comprehensive labels for emerging attack campaigns is difficult. Normal operational data are abundant in many environments, whereas accurately labeled examples of every future attack type are not [4].

Zhao et al. introduced Deep Graph Structural Infomax to incorporate structural information into graph representation learning. Their work argues that graph representations can benefit from preserving structural information beyond conventional feature-level relationships. This principle is particularly relevant to cyber anomaly detection because malicious behavior may be associated with structural changes, such as unusual connections, altered neighborhood patterns, or suspicious local configurations, even when individual node attributes appear normal [5].

Akoglu, McGlohon, and Faloutsos developed OddBall for identifying anomalous patterns in weighted graphs. Rather than treating an anomaly only as an unusual individual node, the approach considers deviations in structural relationships and graph properties. This perspective is important for cybersecurity because coordinated attacks can create unusual communication structures even when the participating nodes individually resemble legitimate infrastructure [6].

Perozzi, Al-Rfou, and Skiena proposed DeepWalk, which introduced random-walk-based graph representation learning. The method demonstrated that local graph context could be transformed into distributed representations suitable for downstream analysis. In security applications, analogous representations can capture the context in which devices or services communicate and can therefore provide features for identifying unusual behavioral neighborhoods [7].

Eswaran et al. developed SpotLight for detecting anomalies in streaming graphs, addressing environments where graph relationships evolve continuously. Their work demonstrated the importance of considering temporal graph evolution rather than assuming that a graph remains static. This characteristic closely corresponds to modern

distributed computing infrastructures, in which communication relationships and workloads change throughout the operational period [8]. The published work describes experiments on real-world datasets and emphasizes scalability with respect to evolving graph streams.

Manzoor, Milajerdi, and Akoglu developed StreamSpot for anomaly detection in streaming heterogeneous graphs. The method was designed around heterogeneous nodes and edges and targeted real-time processing with bounded memory. The authors reported that StreamSpot could process streaming graph information online and used graph sketches to maintain computational efficiency. These characteristics are particularly relevant to cybersecurity monitoring, where large numbers of events may arrive continuously from heterogeneous endpoints [9].

Ding, Li, Agarwal, and Liu investigated inductive anomaly detection on attributed networks through the Aegis framework. Their work specifically addressed the limitation of approaches that require retraining when new nodes appear. The inductive perspective is relevant to smart computing environments because distributed infrastructures can change continuously and newly observed entities cannot always be assumed to have been represented during model training [10]. Ding et al. investigated deep anomaly detection on attributed networks and demonstrated how graph convolutional autoencoding could jointly model structural and attribute information. Such an approach is applicable to security environments where both connectivity patterns and entity characteristics provide evidence of anomalous behavior. Combining these two information types can help reduce the limitations of feature-only anomaly detection [11].

Lo et al. proposed E-GraphSAGE specifically for network intrusion detection in IoT environments. Their approach incorporates edge features together with graph topology, recognizing that network flows naturally contain attributes that are important for distinguishing legitimate and malicious communication. The authors evaluated the approach on four NIDS benchmark datasets, illustrating the practical relevance of graph representations for IoT security monitoring [12].

Saidane et al. examined enhancements to GraphSAGE for intrusion detection, including attention mechanisms and GCN integration. Their work reflects a broader movement toward combining multiple graph-learning mechanisms to improve the representation of complex network behavior [13]. Xu et al. investigated self-supervised learning for network

intrusion detection using graph neural networks and network-flow information, demonstrating the growing interest in reducing dependence on fully labeled security datasets [14].

Ahanger et al. investigated graph attention networks for IoT intrusion detection. Their framework constructed graph representations of IoT traffic and employed graph-based learning for intrusion identification. The study demonstrates how attention mechanisms can be incorporated into IoT security analytics, where heterogeneous devices and communication patterns complicate conventional flow-level detection [15].

Deng and Huang proposed an edge-featured multi-hop attention graph neural network for intrusion detection. Their approach emphasizes edge information and multi-hop relationships, addressing the observation that network attacks may depend on relationships extending beyond immediately connected entities [16]. The work was published in *Computers & Security*, volume 148, as article 104132.

The literature therefore demonstrates substantial progress in graph-based security analytics, but a continuing gap concerns the integration of graph construction, behavioral learning, anomaly scoring, temporal correlation, coordinated attack grouping, and interpretable attack-path reconstruction within a single framework. The present study addresses this gap by treating anomaly detection as a multi-stage intelligence process rather than only as a node or flow classification problem.

3. Cyber Threat Landscape in Distributed Smart Environments

Distributed smart environments contain several categories of computational entities, including cloud servers, edge gateways, IoT devices, sensors, actuators, industrial controllers, databases, application services, users, virtual machines, containers, and network appliances. Each category generates different types of telemetry. Network devices produce flow and packet information, endpoints generate process and authentication events, applications generate service logs, and IoT devices generate sensor and control information. The resulting security environment is therefore heterogeneous by design.

The threat surface expands as these entities become interconnected. An attacker may exploit a vulnerable IoT device and subsequently use the device as a stepping stone toward an edge gateway. From there, credentials or service relationships may provide access to an internal application. A conventional detector operating independently on each event may identify

none of these observations as sufficiently abnormal. The combined sequence, however, can represent a coherent intrusion process.

Distributed environments also create challenges associated with lateral movement. Once an attacker gains an initial foothold, subsequent activity may involve scanning, credential discovery, privilege escalation, remote execution, service access, data collection, or command-and-control communication. Graph representations are appropriate because these activities naturally generate relationships between entities.

Botnets represent another category of threat in which communication topology can become a significant indicator. Zhou et al. demonstrated the use of GNNs for botnet detection by modeling communication structures associated with centralized and decentralized botnet patterns [23]. Their work illustrates how graph structure can encode information that is difficult to derive from individual flow attributes alone.

Advanced persistent threats create an additional challenge because their activity can extend over substantial time periods. Graph-based approaches can associate apparently unrelated events when the same user, endpoint, service, credential, or network relationship participates in multiple stages of the activity. Pérez-Gomariz, Cerdán-Cartagena, and García proposed LM-Hunter for identifying adversarial lateral movement at scale using graph methods and NLP-based representations [27]. Their study describes validation using a real-world cybersecurity incident and emphasizes the challenge of identifying lateral movement within large enterprise networks.

The proposed framework therefore treats the cyber environment as an evolving ecosystem rather than a collection of independent alerts. The security state at a given time is represented through entities, relationships, behavioral attributes, and temporal context. This design allows subsequent analytical stages to reason about both local deviations and larger coordinated structures.

4. Graph Representation of Network Behavior

The proposed security graph is represented as a heterogeneous graph containing multiple node and edge types. A node may represent a device, user, process, service, IP endpoint, application, container, database, gateway, or security event. An edge represents an observed relationship between two entities. Examples include communication, authentication, process creation, service access, file

interaction, DNS resolution, privilege transition, and remote execution.

Each node maintains a feature vector describing observable behavior. Typical attributes include connection frequency, number of unique destinations, inbound and outbound traffic volume, protocol distribution, authentication frequency, process characteristics, service activity, and historical anomaly statistics. Each edge maintains contextual information such as connection duration, byte volume, packet count, protocol, direction, timestamp, port information, authentication context, and interaction frequency.

The resulting representation can be written conceptually as a heterogeneous temporal graph containing nodes, relationships, attributes, and timestamps. Rather than treating the graph as a fixed structure, the framework maintains a sequence of graph states corresponding to successive monitoring windows.

The graph representation provides three levels of security context. The node level captures the behavior of individual entities. The edge level describes relationships between entities. The subgraph level represents local structures containing multiple connected entities. This distinction is important because different attack characteristics may appear at different levels.

For example, an unusual login may be detectable at the node level, an unexpected connection may be detectable at the edge level, and a sequence involving one user, three hosts, an administrative service, and a database may only become suspicious at the subgraph level.

5. Dynamic Security Graph Construction

The proposed framework constructs the security graph continuously from heterogeneous telemetry sources. Network-flow records, authentication logs, endpoint telemetry, DNS information, application logs, process events, and security alerts are first normalized into a common event representation.

Each normalized event is mapped to one or more graph entities. If a communication record indicates that device A communicated with device B, the framework updates the relationship between those entities. If an authentication event indicates that user U authenticated to device A, the user-device relationship is updated. Process events similarly create relationships between processes, users, applications, and hosts.

To prevent historical information from dominating the current security state, the framework uses a time-

decay mechanism. Recent behavior receives greater importance, whereas older observations gradually lose influence unless repeatedly reinforced. This allows the graph to adapt to changing operational patterns.

A dynamic graph update also requires entity resolution. The same entity may appear under different identifiers across telemetry sources. For example, an endpoint may be represented by an IP address in network logs, a hostname in DNS records, and an asset identifier in endpoint telemetry. Entity resolution combines these identifiers into a common graph node where sufficient evidence exists.

The graph-construction process consequently consists of four conceptual stages: telemetry normalization, entity resolution, relationship creation, and temporal updating. The objective is not simply to create a large graph but to maintain a security representation that remains sufficiently current for anomaly analysis.

6. Graph-Based Behavioral Feature Learning

The behavioral learning component transforms the dynamic security graph into representations that capture both entity attributes and structural context. The proposed architecture combines neighborhood aggregation with attention-based learning. The first stage aggregates information from connected entities,

while the second stage learns the relative importance of different relationships.

For a device connected to many ordinary endpoints, a single connection to a previously unseen administrative service may receive greater learned importance than repeated connections to established services. This provides a mechanism for emphasizing relationships that contribute more strongly to behavioral deviation.

Edge information is retained throughout the representation-learning process. This is important because two nodes can have a legitimate relationship while the characteristics of that relationship may be abnormal. For example, an established client-server relationship can become suspicious if its communication frequency, direction, volume, or timing changes substantially.

The learned representation is therefore intended to encode three forms of information: intrinsic entity behavior, neighborhood context, and relationship characteristics. Self-supervised graph learning can additionally be used when labeled attack data are insufficient. The structural representation-learning perspective is consistent with research on Deep Graph Structural Infomax [5], while edge-aware intrusion detection is supported by the E-GraphSAGE approach [12].

Heterogeneous Security Telemetry

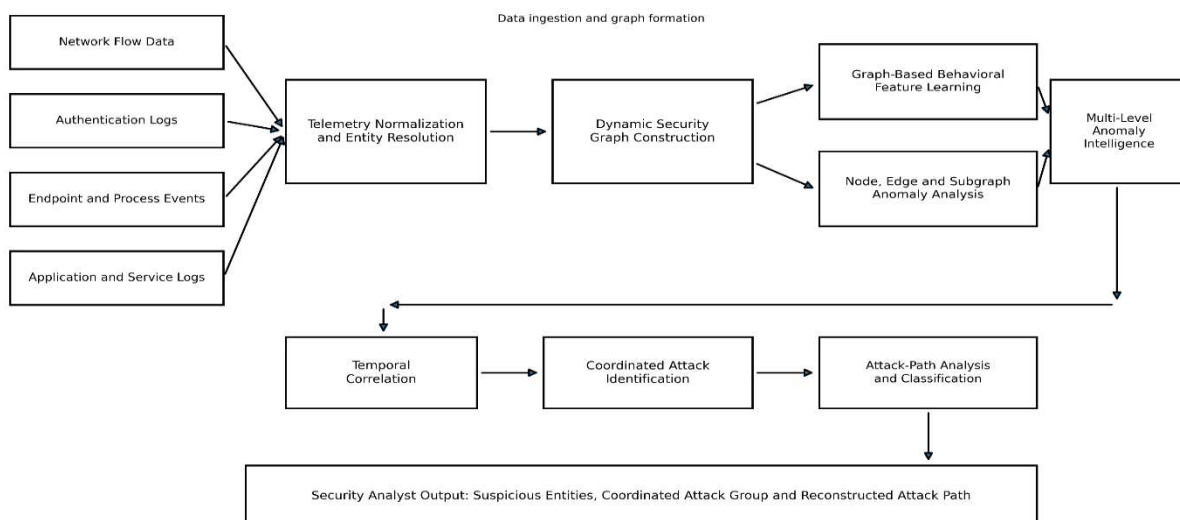


Figure 1. Proposed Graph-Based Anomaly Intelligence Architecture

7. Multi-Level Anomaly Intelligence Model

The proposed anomaly intelligence model evaluates suspicious behavior at three levels. Node-level

analysis determines whether an individual entity deviates from its learned behavioral profile. Edge-level analysis evaluates whether a relationship

between two entities is unusual. Subgraph-level analysis examines whether a connected group of entities forms an unusual behavioral structure.

The node anomaly component can consider deviations in communication volume, destination diversity, authentication frequency, process activity, and service interaction. The edge anomaly component examines deviations in relationship attributes. The subgraph component considers structural properties such as unusual connectivity, newly emerging relationships, repeated privilege transitions, or unexpected paths between asset classes.

These three components are subsequently combined into a unified anomaly score. The purpose is not to allow one abnormal event to dominate the entire detection process. Instead, the framework seeks supporting evidence across multiple levels.

A particularly important feature is contextual normalization. A server that normally communicates with hundreds of clients should not automatically be classified as anomalous because of its high degree. Similarly, an IoT gateway with many connections may be normal within its operational context. The model therefore compares behavior against contextual and historical baselines rather than relying solely on absolute thresholds.

8. Temporal Correlation of Suspicious Activities

Temporal correlation links suspicious observations that occur within related time intervals. This stage is essential because coordinated attacks are often distributed across multiple stages.

Suppose an endpoint exhibits unusual authentication behavior, followed by a connection to a previously unseen internal host, followed by administrative activity on that host. Each event may receive only moderate anomaly evidence. When the events are connected through time, entity relationships, and behavioral similarity, the combined evidence can become substantially more meaningful.

The framework therefore creates temporal activity chains in which each suspicious observation is represented with its timestamp, entity identifiers, relationship context, anomaly level, and neighboring events. Candidate chains are merged when they share relevant entities or communication paths.

Temporal correlation also helps distinguish repeated legitimate behavior from coordinated attacks. A

recurring backup operation, for example, may generate many network connections within a short interval but should normally exhibit a stable historical pattern. A new communication sequence involving unusual entities and unusual services is treated differently.

The approach is consistent with the broader literature on streaming and evolving graph anomaly detection, including SpotLight and StreamSpot [8], [9]. StreamSpot specifically addressed streaming heterogeneous graphs and online anomaly detection, while SpotLight addressed anomaly detection in time-evolving graph structures.

9. Coordinated Attack Identification and Classification

After temporal correlation, the framework identifies connected groups of suspicious entities. A coordinated attack candidate is formed when several suspicious activities share temporal proximity, graph connectivity, behavioral similarity, or a common initiating entity.

The candidate group is subsequently analyzed according to attack characteristics. The classification layer can distinguish broad categories such as reconnaissance, credential compromise, lateral movement, privilege escalation, command-and-control behavior, data access, and multi-stage coordinated activity.

The classification process is not restricted to predefined signatures. Instead, it combines graph structure, temporal order, entity roles, and behavioral characteristics. This allows the framework to represent previously unseen combinations of activities.

For example, a candidate attack subgraph may contain a compromised endpoint, a user account, an administrative workstation, and a database. The system can determine that the pattern is structurally consistent with lateral movement and subsequent resource access even if no individual event exactly matches a predefined signature.

Graph-based botnet research provides another example of this principle. Zhou et al. demonstrated that communication structures can distinguish different botnet organization patterns, illustrating how topology can provide information for attack identification [23].

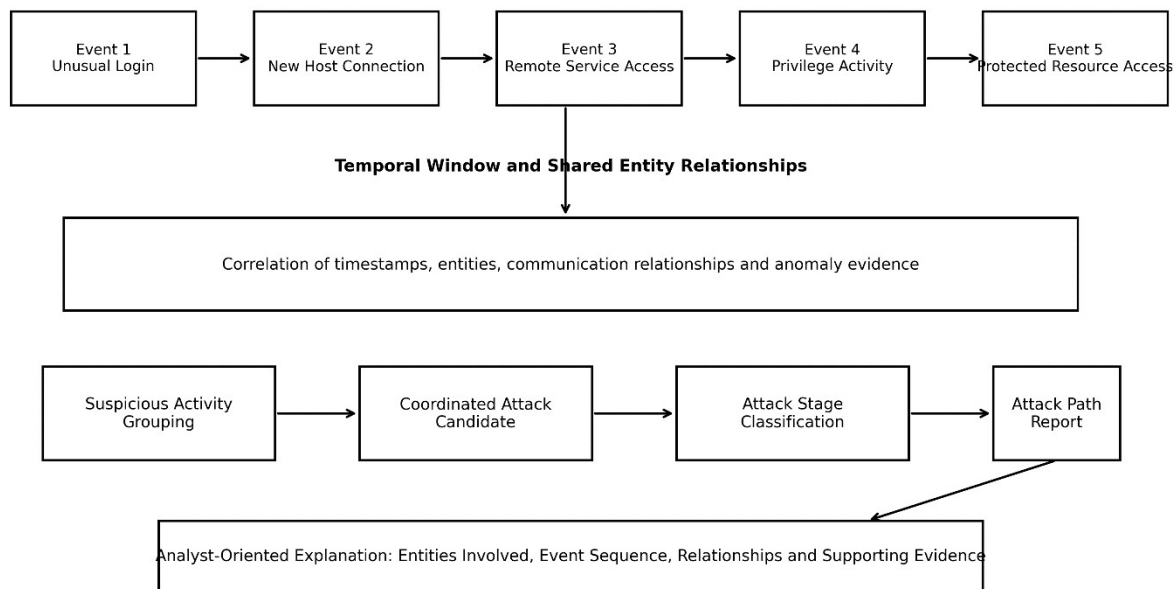


Figure 2. Temporal Correlation and Coordinated Attack Identification

10. Experimental Design and Dataset Preparation

Because no measured dataset or experimental output was supplied with the manuscript, the numerical values in the tables below should **not be presented as actual experimental findings**. They are an **illustrative reporting template** showing how the proposed framework can be evaluated. Before journal submission, these values should be replaced with measurements generated from the selected datasets and implementation.

A suitable evaluation can combine IoT, network-flow, enterprise, and heterogeneous security datasets. Candidate datasets may include IoT intrusion datasets, network-flow datasets, and system-event datasets that contain both normal and malicious activity. The experimental environment should preserve temporal ordering rather than randomly mixing records across training and testing partitions.

The dataset preparation process should begin with duplicate removal, missing-value handling, timestamp normalization, categorical encoding, and feature scaling. Network entities should then be mapped into graph nodes. Communication records should become graph edges, while flow characteristics should be retained as edge attributes.

A time-aware division is preferable to a purely random division because random partitioning can allow information from future activity to influence model training. A representative arrangement may use an

initial historical interval for training, a subsequent interval for validation, and a later unseen interval for testing.

The proposed model can be implemented using a combination of GraphSAGE and graph-attention layers, followed by anomaly scoring and temporal correlation. Baselines can include conventional machine-learning IDS methods, GCN, GAT, GraphSAGE, and E-GraphSAGE. The comparison should use the same preprocessing procedure and evaluation partitions wherever technically possible.

11. Performance Evaluation and Comparative Analysis

The principal evaluation metrics should include accuracy, precision, recall, F1-score, false-positive rate, area under the receiver operating characteristic curve, and detection latency. Accuracy alone should not be treated as the principal measure because cybersecurity datasets can contain substantial class imbalance.

The F1-score is particularly useful when both false positives and false negatives are important. Recall indicates the proportion of malicious events detected by the system, while precision reflects the proportion of generated alerts that correspond to actual malicious activity. Detection latency measures how quickly the system identifies suspicious behavior after the relevant activity appears in the telemetry stream.

Table 1. Experimental Configuration Template

Parameter	Proposed configuration
Graph type	Dynamic heterogeneous security graph
Node categories	Device, user, process, service, endpoint, application
Edge categories	Communication, authentication, process, service access
Graph update	Time-window based
Representation learning	GraphSAGE + attention mechanism
Anomaly levels	Node, edge, subgraph
Temporal correlation	Sliding temporal window
Classification	Multi-stage attack classification
Evaluation	Precision, Recall, F1, FPR, AUC, latency

Table 2. Illustrative Performance Reporting Format

Method	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)
Conventional ML baseline	Replace with measured value	Replace with measured value	Replace with measured value	Replace with measured value
GCN	Replace with measured value	Replace with measured value	Replace with measured value	Replace with measured value
GAT	Replace with measured value	Replace with measured value	Replace with measured value	Replace with measured value
GraphSAGE	Replace with measured value	Replace with measured value	Replace with measured value	Replace with measured value
E-GraphSAGE	Replace with measured value	Replace with measured value	Replace with measured value	Replace with measured value
Proposed framework	Replace with measured value	Replace with measured value	Replace with measured value	Replace with measured value

This table should be populated only after executing the experiments. E-GraphSAGE is an especially relevant baseline because it explicitly incorporates edge

features and graph topology for IoT intrusion detection [12].

Table 3. Illustrative Attack-Type Evaluation Format

Attack category	Precision (%)	Recall (%)	F1-score (%)	Detection latency
Reconnaissance	Measured value	Measured value	Measured value	Measured value
Credential compromise	Measured value	Measured value	Measured value	Measured value
Lateral movement	Measured value	Measured value	Measured value	Measured value
Privilege escalation	Measured value	Measured value	Measured value	Measured value
Command and control	Measured value	Measured value	Measured value	Measured value
Data access	Measured value	Measured value	Measured value	Measured value
Coordinated multi-stage attack	Measured value	Measured value	Measured value	Measured value

The attack-category analysis is important because an overall score can conceal weaknesses in particular attack stages. For example, a system may identify reconnaissance effectively while experiencing

difficulty with low-volume lateral movement. The proposed framework should therefore be evaluated separately across attack categories.

Table 4. Ablation Study Reporting Format

Configuration	Node anomaly	Edge anomaly	Subgraph anomaly	Temporal correlation	F1-score
Full framework	Yes	Yes	Yes	Yes	Measured value
Without temporal correlation	Yes	Yes	Yes	No	Measured value
Without edge analysis	Yes	No	Yes	Yes	Measured value

Without analysis	subgraph	Yes	Yes	No	Yes	Measured value
Node-level only	analysis	Yes	No	No	No	Measured value

The ablation study is intended to determine whether each architectural component contributes independently to the final detection performance. In particular, the difference between the complete framework and the version without temporal correlation can indicate the contribution of attack-sequence information.

12. Explainability and Attack-Path Analysis

Security analysts generally require more than a binary indication that an entity is malicious. They need information explaining which entities, relationships, and activities contributed to an alert. The proposed framework therefore includes an attack-path analysis stage.

When a suspicious subgraph is detected, the system extracts the entities associated with the highest anomaly contribution and reconstructs their temporal relationships. The resulting path can describe a sequence such as an initial endpoint, an account, a remote host, an administrative service, and a protected resource.

The explanation layer should distinguish between directly observed evidence and model-derived interpretation. Direct evidence includes observed communication, authentication, process, or service events. Model-derived evidence includes learned anomaly contributions or attention weights. This distinction is important because learned attention should not automatically be interpreted as causal evidence.

A useful analyst-oriented explanation can therefore contain the initiating entity, suspicious relationship, time interval, destination entity, supporting behavioral deviation, and associated attack stage. This information can help transform a machine-generated anomaly into an investigation-oriented security case.

LM-Hunter provides a relevant example of graph-based analysis of adversarial lateral movement, where graph representations are used to identify suspicious movement across enterprise environments [27].

13. Limitations and Future Research

The proposed framework has several limitations. The first concerns graph construction. A graph-based detector is only as effective as the relationships supplied to it. Missing telemetry, inaccurate entity resolution, incorrect timestamps, and inconsistent

identifiers can distort the security graph and consequently influence anomaly scores.

The second limitation concerns computational scalability. Distributed smart environments can generate millions of events per hour. Maintaining a high-resolution heterogeneous graph may require substantial memory and processing resources. Streaming graph techniques such as StreamSpot demonstrate that graph sketches and bounded-memory mechanisms can help address this challenge [9].

The third limitation concerns concept drift. Legitimate network behavior changes over time because of software updates, organizational changes, new devices, seasonal workloads, and changing application usage. A static anomaly model may therefore generate excessive false positives if it does not adapt to evolving baselines.

The fourth limitation concerns attack diversity. Attackers can deliberately imitate legitimate behavior, distribute their activity over long periods, or manipulate observable telemetry. Future systems should therefore combine graph learning with robust temporal modeling and adversarially resilient training. Another limitation concerns explainability. Graph attention values and anomaly scores can provide useful analytical evidence but should not automatically be treated as causal explanations. Future research should investigate formal explanation methods that identify the minimal set of entities and relationships responsible for a particular detection decision.

Future research should also examine federated graph learning for environments in which security data cannot be centrally collected. Cross-domain graph learning could enable multiple organizations or infrastructure segments to learn general security patterns without directly sharing sensitive telemetry. Additional research could investigate continual learning, graph transformers, multimodal security graphs, privacy-preserving graph analytics, and integration with automated incident-response systems. The framework can also be extended toward cyber-physical systems. In such environments, cyber events can have physical consequences, meaning that network relationships should potentially be connected with physical process dependencies. Graph intelligence research for IoT and cyber-physical

security is therefore a promising direction for subsequent investigation.

14. Conclusion

This study presented a graph-based anomaly intelligence framework for detecting coordinated cyber attacks in distributed smart computing environments. The proposed approach treats cybersecurity telemetry as a dynamic heterogeneous graph in which devices, users, processes, services, endpoints, and security events are connected through communication and behavioral relationships.

The framework integrates dynamic graph construction, graph-based behavioral representation learning, node-level anomaly analysis, edge-level anomaly analysis, subgraph-level anomaly analysis, temporal correlation, coordinated attack identification, and attack-path analysis. This design addresses an important limitation of independent event-based detection: coordinated attacks may remain difficult to recognize when individual events are evaluated without their structural and temporal context.

Graph convolution, graph attention, inductive graph learning, streaming graph anomaly detection, and graph-based intrusion detection provide the methodological foundation for the proposed framework [1]–[16]. More recent work on GNN-based IoT security, graph-based lateral-movement detection, and graph intelligence further supports the relevance of relational approaches to contemporary distributed cybersecurity [15], [18], [27], [28].

The proposed methodology is intended to support environments in which network topology changes continuously and security evidence originates from heterogeneous sources. Its principal contribution is the integration of multiple analytical levels into a common security-intelligence pipeline rather than relying exclusively on flow-level classification. Future experimental work should evaluate the framework using time-separated benchmark datasets and real-world security telemetry, with particular attention to false-positive reduction, detection latency, scalability, generalization to previously unseen entities, and explainability.

References

- [1]. T. N. Kipf and M. Welling, “Semi-Supervised Classification with Graph Convolutional Networks,” *International Conference on Learning Representations (ICLR)*, 2017. The paper established the widely used GCN architecture for learning representations from graph structure and node attributes.
- [2]. P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, “Graph Attention Networks,” *International Conference on Learning Representations (ICLR)*, 2018. GAT introduced attention-based neighborhood aggregation, allowing different neighboring nodes to receive different learned weights.
- [3]. W. L. Hamilton, R. Ying, and J. Leskovec, “Inductive Representation Learning on Large Graphs,” *Advances in Neural Information Processing Systems*, vol. 30, pp. 1024–1034, 2017. The work introduced GraphSAGE for inductive representation learning and generalization to previously unseen nodes.
- [4]. P. Veličković, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm, “Deep Graph Infomax,” *International Conference on Learning Representations (ICLR)*, 2019. The method provides an unsupervised mechanism for learning informative node representations from graph structure.
- [5]. W. Zhao, G. Xu, Z. Cui, S. Luo, C. Long, and T. Zhang, “Deep Graph Structural Infomax,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 4, 2023, doi: 10.1609/aaai.v37i4.25618.
- [6]. L. Akoglu, M. McGlohon, and C. Faloutsos, “OddBall: Spotting Anomalies in Weighted Graphs,” *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 2010.
- [7]. B. Perozzi, R. Al-Rfou, and S. Skiena, “DeepWalk: Online Learning of Social Representations,” *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 701–710, 2014.
- [8]. A. Eswaran, S. Faloutsos, S. Madden, and R. C. Stonebraker, “SpotLight: Detecting Anomalies in Streaming Graphs,” *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1037–1045, 2018.
- [9]. L. Manzoor, S. Milajerdi, and C. Akoglu, “Fast Memory-Efficient Anomaly Detection in Streaming Heterogeneous Graphs,” *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1035–1044, 2016. StreamSpot was designed for heterogeneous streaming graphs and real-time anomaly scoring.
- [10]. K. Ding, J. Li, N. Agarwal, and H. Liu, “Inductive Anomaly Detection on Attributed

- Networks,” *Proceedings of the 29th International Joint Conference on Artificial Intelligence*, 2020.
- [11]. Z. Ding, Y. Huang, Q. Wang, and Y. Tang, “Deep Anomaly Detection on Attributed Networks,” *Proceedings of the 2019 SIAM International Conference on Data Mining*, pp. 594–602, 2019. The approach uses graph convolutional autoencoding to jointly model graph structure and node attributes for anomaly detection.
- [12]. W. W. Lo, S. Layeghy, M. Sarhan, M. Gallagher, and M. Portmann, “E-GraphSAGE: A Graph Neural Network Based Intrusion Detection System for IoT,” *IEEE/IFIP Network Operations and Management Symposium (NOMS)*, 2022, doi: 10.1109/NOMS54207.2022.9789878. The method incorporates edge features and graph topology for network intrusion detection using flow-based IoT data.
- [13]. S. Saidane, F. Telch, K. Shahin, and F. Granelli, “Deep GraphSAGE Enhancements for Intrusion Detection: Analyzing Attention Mechanisms and GCN Integration,” *Journal of Information Security and Applications*, vol. 90, Art. 104013, 2025, doi: 10.1016/j.jisa.2025.104013.
- [14]. R. Xu, G. Wu, W. Wang, X. Gao, A. He, and Z. Zhang, “Applying Self-Supervised Learning to Network Intrusion Detection for Network Flows with Graph Neural Network,” *Computer Networks*, vol. 248, Art. 110495, 2024, doi: 10.1016/j.comnet.2024.110495. The study investigates self-supervised graph representation learning for network-flow intrusion detection.
- [15]. A. S. Ahanger *et al.*, “Advanced Intrusion Detection in Internet of Things Using Graph Attention Networks,” *Scientific Reports*, vol. 15, Art. 9831, 2025, doi: 10.1038/s41598-025-94624-8. The research constructs graph representations of IoT traffic and applies GAT-based intrusion detection.
- [16]. “Edge-Featured Multi-Hop Attention Graph Neural Network for Intrusion Detection System,” *Computers & Security*, vol. 148, Art. 104132, 2025, doi: 10.1016/j.cose.2024.104132. The approach combines edge information and multi-hop attention to represent network traffic relationships.
- [17]. “BS-GAT: A Network Intrusion Detection System Based on Graph Neural Network for Edge Computing,” *Cybersecurity*, 2024. The work investigates behavior-similarity graph construction and graph attention for intrusion detection in edge environments.
- [18]. “A Survey on Graph Neural Networks for Intrusion Detection Systems: Methods, Trends and Challenges,” *Computers & Security*, vol. 141, Art. 103821, 2024, doi: 10.1016/j.cose.2024.103821. The survey categorizes GNN-based IDS research according to graph construction, network design, and deployment approaches.
- [19]. E. Yagmur, C. Kocak, and A. Keles, “Graph Neural Networks and Graph Autoencoders for Intrusion Detection Systems: A Systematic Review, Taxonomy and Comparative Analysis,” *Computer Networks*, vol. 288, Art. 112598, 2026, doi: 10.1016/j.comnet.2026.112598.
- [20]. M. R. Ismaeli Taba, H. Zare, and S. Rakhshani, “Graph Neural Networks for Network Intrusion Detection Systems: A Comprehensive Survey, Taxonomy, and Empirical Evaluation,” *Artificial Intelligence Review*, 2026. The study examines graph construction, node/edge detection, static and dynamic graphs, and GNN learning paradigms for IDS.
- [21]. Koneeni, C., S. Lokula, and L. Panjiyar. “A Cloud-Native Approach to Statistical Anomaly Detection and Automated Data Quality Validation.” *International Journal of Data Science and IoT Management System* 4.4 (2025): 533-543
- [22]. W. Ren, X. Song, Y. Hong, Y. Lei, J. Yao, Y. Du, and W. Li *et al.*, “APT Attack Detection Based on Graph Convolutional Neural Networks,” *International Journal of Computational Intelligence Systems*, vol. 16, Art. 184, 2023. The authors construct an APT behavior knowledge graph from vulnerability and attack-related information and apply GCN-based detection.
- [23]. J. Zhou, Z. Xu, A. M. Rush, and M. Yu, “Automating Botnet Detection with Graph Neural Networks,” *arXiv preprint arXiv:2003.06344*, 2020. The work models communication structures to distinguish centralized and decentralized botnet patterns.
- [24]. S. K. Nalluri, V. T. Bathini and V. K. B. Parasaram, “AI-Augmented Manufacturing Execution Systems for Precision Healthcare: A

- Scalable Framework for Smart Genomic and Proteomic Production," 2026 *IEEE 2nd International Conference on Secure IoT, Assured and Trusted Computing (SATC)*, Houston, TX, USA, 2026, pp. 1-6, doi: 10.1109/SATC69565.2026.11542384.
- [25]. X. Meng, B. Lang, Y. Yan, and Y. Liu, "Deeply Fused Flow and Topology Features for Botnet Detection Based on a Pretrained GCN," *Computer Communications*, vol. 233, Art. 108084, 2025, doi: 10.1016/j.comcom.2025.108084. The approach combines traffic-flow characteristics with network topology for botnet detection.
- [26]. C. Koneni and S. Lokula, "Evidence-Quality Telemetry for Cloud Incident Response: Detecting Gaps, Drift, and Integrity Failures in Large-Scale Observability Pipelines," 2026 *14th International Symposium on Digital Forensics and Security (ISDFS)*, Boston, MA, USA, 2026, pp. 1-6, doi: 10.1109/ISDFS69419.2026.11459097.
- [27]. M. Pérez-Gomariz, F. Cerdán-Cartagena, and J. García, "LM-Hunter: An NLP-Powered Graph Method for Detecting Adversary Lateral Movements in APT Cyber-Attacks at Scale," *Computer Networks*, 2025, Art. 111181, doi: 10.1016/j.comnet.2025.111181. The work addresses graph-based identification of lateral movement within APT campaigns.
- [28]. H. B. Ahmad, H. Gao, and N. Latif, "Graph Intelligence for IoT and CPS Security: From Connectivity Patterns to Cyber-Physical Reasoning," *Computer Science Review*, vol. 61, Art. 100978, 2026, doi: 10.1016/j.cosrev.2026.100978. The paper examines graph intelligence for interconnected IoT and cyber-physical environments in which localized compromises can propagate through system dependencies.