

Article

Provenance Graph Modeling and Feature Enhancement for Power System APT Detection

Xuan Zhang, Haohui Su *, Lincheng Li and Lvjun Zheng

Extra High Voltage Power Transmission Company, China Southern Power Grid Co., Ltd.,
Guangzhou 510663, China; zhangxuan1@ehv.csg.cn (X.Z.); lilincheng@ehv.csg.cn (L.L.);
zhenglvjun@ehv.csg.cn (L.Z.)

* Correspondence: suhaohui@ehv.csg.cn; Tel.: +86-18825136393

Abstract

The power system, as a critical national infrastructure, faces stealthy and persistent intrusions from Advanced Persistent Threat (APT) attacks. These attack chains span multiple stages and components, while heterogeneous data sources lack unified semantics, limiting the interpretability of current detection methods. To address this, we combine the W3C PROV-DM standard with power-specific semantics to map generic provenance data into standardized provenance graphs. On this basis, we propose a graph neural network framework that jointly models temporal dependencies and structural features. The framework constructs unified provenance graphs with snapshot partitioning, applies Functional Time Encoding (FTE) for temporal modeling, and employs a graph attention autoencoder with node masking and edge reconstruction to enhance feature representations. Through pooling, graph-level embeddings are obtained for downstream detection. Experiments on two public datasets show that our method outperforms baselines across multiple metrics and exhibits clear inter-class separability. In the context of scarce power-domain APT data, this study improves model applicability and interpretability, and it provides a practical path for provenance graph-based intelligent detection in critical infrastructure protection.

Keywords: APT attacks; graph neural networks; power system security; graph representation learning; feature enhancement



Academic Editor: Gianpaolo Vitale

Received: 17 September 2025

Revised: 19 October 2025

Accepted: 26 October 2025

Published: 29 October 2025

Citation: Zhang, X.; Su, H.; Li, L.; Zheng, L. Provenance Graph Modeling and Feature Enhancement for Power System APT Detection. *Electronics* **2025**, *14*, 4241. <https://doi.org/10.3390/electronics14214241>

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

As a critical component of national infrastructure, the power system plays a vital role in national security and public interests. With the transition toward “intelligent, informatized, and networked” operation, traditional physical facilities are increasingly integrated with modern information and communication technologies (ICTs). All key processes—including generation, transmission, distribution, dispatching, and end-user consumption—now rely on network communication. While this transformation greatly improves automation and intelligence, it also introduces significant cybersecurity risks.

Attackers can exploit information systems to remotely interfere with physical operations, causing severe damage. Advanced Persistent Threats (APTs), known for their stealthiness, persistence, and targeted nature, are regarded as one of the most destructive attack forms against power systems. Typically launched by well-resourced and highly organized actors (e.g., nation-state groups), APTs penetrate through external entry points such as office terminals, supply chain software, or VPN gateways, then progressively infiltrate the control domain via lateral movement and privilege escalation, ultimately

compromising core assets. Once successful, APTs often inflict catastrophic impacts on critical infrastructure. For instance, the notorious Stuxnet worm infiltrated Iran's Natanz nuclear facility via USB media and destroyed about 1000 uranium centrifuges, significantly hindering its nuclear program. Similarly, Ukraine's power grid suffered large-scale black-outs in 2015 and 2016 due to APT campaigns by the Russian Sandworm group, leaving more than 230,000 users without electricity for hours.

Unlike conventional intrusions, APTs are highly targeted, sophisticated, long-lived, and difficult to detect, often being discovered only after tangible damage occurs. As a result, traditional passive defense strategies—based primarily on post-event detection and isolation—struggle to identify and block such covert threats in time.

To enable the in-depth modeling and effective attribution of APT behaviors, recent studies have increasingly focused on graph-based methods for attack tracking. In highly coupled and process-driven systems such as power grids, APT chains exhibit clear temporal sequences and causal structures, making provenance graphs a powerful tool for visualization and correlation analysis. Provenance graphs represent system objects (e.g., processes, files, and network connections) as nodes and causal or dependency relations as edges, providing structured and traceable representations. Compared with log scanning or rule-based detection, provenance graphs offer the following three major advantages: (i) they naturally capture the chain-like behaviors of APTs, enabling the reconstruction of complete attack paths; (ii) they integrate heterogeneous data sources while preserving contextual information, revealing hidden correlations; and (iii) they support advanced representation learning via graph neural networks (GNNs), which jointly model attributes and structures to improve detection accuracy and generalization. Consequently, provenance graphs have emerged as an essential tool for APT detection and forensic analysis in industrial control and critical infrastructure systems.

However, provenance-based APT analysis still faces several challenges, outlined as follows:

1. APT-related raw data are often recorded in heterogeneous formats (e.g., logs, tables, and PCAP traffic), lacking unified semantics or structural standards, which complicates graph construction and introduces ambiguity.
2. Most public datasets originate from generic operating systems or enterprise networks, lacking the realistic characteristics of power systems, which hinders domain-specific training and validation.
3. APT behaviors exhibit strong temporal dependencies, yet existing work mostly focuses on structural modeling, with insufficient attention to event dynamics, making it difficult to capture stealthy or progressive attack behaviors.
4. Current feature representations for nodes and graphs remain coarse, lacking fine-grained semantic modeling and dynamic expression. In highly covert scenarios, existing features provide limited discriminative power, restricting detection performance.

Therefore, this paper aims to address the aforementioned challenges by constructing a unified provenance graph analysis framework for APT detection within the context of power systems, thereby enhancing the system's capability for proactive threat perception and defense against stealthy attacks. The main contributions are summarized as follows:

1. Based on the W3C PROV-DM standard and combined with the operational characteristics of power systems, we design a semantic mapping and standardized modeling method for multi-source security data, ensuring the interpretability and consistency of APT behaviors in power-related scenarios.
2. Using the CICAPT-IIoT dataset, event-level logs are parsed and transformed into temporal snapshots. Through the integration of One-Hot and Functional Time Encoding

mechanisms, both entity-type features and temporal dependencies are jointly modeled to capture the staged and latent characteristics of APT attacks.

3. A node-masking and edge-reconstruction mechanism is incorporated into a graph attention autoencoder, which, together with feature enhancement and a lightweight downstream detection algorithm, enables the effective identification of anomalous behaviors. Experiments conducted on the processed CICAPT-IIoT and Unicorn Wget datasets validate the effectiveness of each module, demonstrating that the proposed method achieves superior performance in APT attack detection.

2. Background

2.1. Characteristics of APT Attacks

Unlike conventional malware or opportunistic intrusions, an Advanced Persistent Threat (APT) does not rely on short-term brute-force destruction but instead follows a strategy of precise infiltration, prolonged persistence, and targeted control to achieve deep and sustained dominance over specific victims. Essentially, it represents a strategic cyber-espionage campaign conducted by well-resourced, highly organized actors—often driven by political, military, or economic objectives.

From an operational perspective, APTs typically follow a multi-stage and modular kill chain, encompassing reconnaissance, intrusion, persistence, lateral movement, command and control (C2), and final exploitation or disruption.

The “advanced” aspect refers to the attackers’ use of sophisticated techniques—such as zero-day exploits, social engineering, supply chain compromise, encrypted tunnels, and multi-layer proxy control—to penetrate defenses. APT operators possess detailed knowledge of the target architecture and can evade firewalls, intrusion detection systems, and access controls through multi-vector infiltration, often traversing from IT and management networks down to control systems and field devices while deploying customized payloads and protocols at each stage.

The “persistent” nature of APTs lies in their long-term concealment and adaptive control. Instead of immediate destruction, adversaries maintain stealthy access for extended periods by periodically updating payloads, masquerading as legitimate processes, or exploiting redundancy to preserve privileges. When defenses evolve, they dynamically adjust tactics or switch backdoor channels to ensure survival.

The “threat” dimension reflects the mission-oriented and deeply intrusive intent of APTs. Attackers typically target high-value assets—such as critical infrastructures, energy grids, research institutions, or military communication networks—with goals extending beyond financial gain to include functional disruption, control manipulation, and strategic data exfiltration.

Another defining feature is stealth and anti-forensic capability. To evade detection, APTs employ code obfuscation, in-memory execution, staged activation, encrypted communications, and traffic camouflage. Some attacks forge digital signatures or inject into legitimate processes to mimic normal system behavior, defeating signature-based defenses. Moreover, they often erase logs, alter timestamps, manipulate alarms, or fabricate traffic patterns, severely hindering post-incident forensics and attribution.

In contrast, traditional malware tends to rely on single-vector intrusion and short-term impact, whereas APTs achieve continuous, occupation-style control through multi-path infiltration, distributed command channels, and resilient communication structures.

2.2. Impacts and Causes of APT on Power Systems

According to [1], the destructive impact of Advanced Persistent Threats (APTs) on power systems typically unfolds in two distinct stages. In the first stage, characterized

by long-term infiltration and reconnaissance, attackers penetrate enterprise networks via phishing emails, software vulnerabilities, or supply chain compromises. During this phase, they stealthily collect critical information such as system topology, device configurations, communication protocols, and access credentials. In the second stage, attackers transition to control exploitation and destructive execution. By forging or tampering with IEC 61850 [2] communication messages—such as the GOOSE, SV, and MMS packets—they inject malicious commands to manipulate circuit breakers, protective relays, and control logic. Experimental results demonstrate that such attacks can circumvent conventional firewalls and intrusion detection systems, directly inducing protection malfunctions, time synchronization anomalies, and even physical damage to power equipment. For instance, by altering the Precision Time Protocol (PTP), adversaries can cause measurement misalignment and command sequence disorder, thereby disrupting relay protection and automatic reclosing mechanisms, ultimately leading to false tripping or load imbalance.

From a systemic perspective, the cyber–physical coupling inherent in power grids amplifies the consequences of APT attacks compared to those in pure IT environments. Once the station control layer is compromised, attackers can exploit the trust chain between the control center and the process layer to inject malicious or falsified data—known as False Data Injection (FDI)—thereby deceiving state estimation models and manipulating protection logic. This may trigger erroneous dispatch commands, false tripping, or even cascading failures. Due to the complex dynamic interconnections and redundancy in modern grids, localized disruptions can propagate through the network topology and evolve into wide-area instabilities. Studies have shown that coordinated attacks against multiple substations within a short time window could lead to large-scale system instability or blackouts.

More stealthy forms of attack involve compromising the supply chain and firmware layers. APT adversaries can implant malicious code into intelligent meters, control gateways, or Intelligent Electronic Device (IED) firmware to continuously exfiltrate data or await specific activation conditions. Once triggered, such backdoors can simultaneously disrupt multiple substations, causing wide-area failures or communication outages.

Even more critically, APT actors often erase logs and alarms during the execution phase, effectively concealing traces of the intrusion and significantly complicating post-incident forensics and response efforts.

The susceptibility of modern power systems to APT attacks arises primarily from their increasing levels of intelligence, interconnectivity, and openness, which have dramatically expanded the system's attack surface. Unlike traditional Information Technology (IT) infrastructures, modern digital substations following the IEC 61850 standard exhibit a vertically integrated architecture where APTs can penetrate from the data and control layers down to the physical layer, potentially leading to blackouts, equipment damage, and system-wide instability.

IEC 61850-based digital substations adopt a hierarchical architecture comprising a control layer for system monitoring and command dispatch, a station control layer linking IEDs and gateways, and a process layer transmitting high-frequency SV and GOOSE messages synchronized via IEEE 1588 PTP [3]. These layers interconnect through Ethernet and TCP/IP networks, forming an end-to-end digital chain that, while improving automation and precision, also introduces significant security vulnerabilities.

Industrial control protocols such as IEC 61850 and IEC 60870-5-104 [4], designed primarily for interoperability and low latency, lack intrinsic mechanisms for authentication, encryption, and data integrity verification—thereby enabling attackers to manipulate or replay control messages. In addition, many substations still operate legacy devices without adequate protection capabilities, providing convenient footholds for persistent

backdoors and lateral movement. The stringent real-time and low-latency requirements of power grids also limit the deployment of conventional deep-packet inspection or flow-monitoring mechanisms.

Moreover, APT campaigns are low-frequency, low-noise, and long-dwelling by nature, making them difficult to detect through traditional intrusion or traffic monitoring systems. To maintain millisecond-level control response, substations often minimize packet inspection, encryption, and multi-layer gateway defenses—conditions that further obscure the subtle, progressive patterns of APT behavior within operational noise. Consequently, when the destructive phase is finally activated, defenders often find that it is already too late.

3. Related Work

3.1. Modeling and Representation

3.1.1. Standardized Construction

In the tracking and analysis of APT attacks, a fundamental challenge lies in constructing clear and provenance-ready graph representations from heterogeneous data such as system logs and network traffic. To this end, many studies have focused on unified provenance standards and methods for multi-source data transformation.

The W3C PROV-DM provenance data model [5] provides a unified theoretical foundation for modeling APT provenance graphs. It defines core elements—Entity, Activity, and Agent—along with standardized relations such as generation, usage, derivation, and association, thereby offering a consistent framework for the causal modeling of attack chains and the standardization of heterogeneous data integration. Numerous works have been developed around this standard. Missier et al. proposed the D-PROV model [6], which extends PROV-DM with workflow structure modeling to express sequential dependencies across attack stages. They later introduced the ProvAbs tool [7], which enables the policy-driven abstraction of provenance graphs to hide sensitive nodes while preserving causal consistency, making it suitable for privacy-sensitive domains such as power systems. Firth and Missier developed ProvGen [8], a system that generates synthetic provenance graphs with customizable structures, supporting attack simulation, feature modeling, and model testing. Wittner et al. [9] designed a lightweight distributed provenance model for real-world multi-institutional environments, extending the applicability of PROV-DM across systems and domains. In addition, Sembay et al. [10] proposed a method to instantiate PROV graphs directly from raw logs in healthcare systems, demonstrating how the standard can be applied in practice.

Yusof et al. introduced the PROVCON framework [11] to construct APT provenance graphs that accurately reflect real-world cyberattacks. Their approach extracts attack primitives from the latest Cyber Threat Intelligence (CTI) reports, categorizes them into environment and event descriptions, and translates them into code for deployment in cyber ranges. By reproducing attacks and collecting heterogeneous data such as system logs, network traffic, and memory dumps, PROVCON generates provenance graphs with automated annotation based on attack descriptions, producing APT graphs enriched with timely and comprehensive indicators.

Kapoor et al. proposed the Flurry framework [12], which simulates web attacks in virtual machines, capturing kernel-level logs via CamFlow and application-level logs via custom hooks. Their CF2G tool then transforms these heterogeneous sources into unified, structured provenance graphs. The framework supports outputs in formats such as NetworkX and Deep Graph Library, enabling direct use in graph machine learning and providing a foundation for anomaly detection based on provenance graphs.

3.1.2. Structural Optimization and Simplification

APT attack chains are often accompanied by complex intermediate events, frequent invocations, and intensive access behaviors. The direct mapping of such activities into provenance graphs typically results in node explosion, redundant paths, and dependency inflation, which hinder representation learning and reduce detection efficiency. Therefore, graph compression and structural optimization are essential.

Li et al. proposed ProvGRP [13], which addresses the errors and redundant events caused by coarse-grained logs. Their method introduces context-based behavior partitioning and path merging to improve investigation efficiency, serving as a critical preprocessing step for provenance-based APT detection. By optimizing graph structures, ProvGRP enhances both the efficiency and reliability of detection.

Men et al. introduced GETSUS [14], a processor-tracing (PT)-based approach for constructing efficient provenance graphs with hardware support. GETSUS targets the “dependency explosion” and missing critical event sequences in traditional provenance graphs. It compresses graphs with thousands or even millions of edges into graphs with only tens of edges, while retaining the core semantics of attack flows. This achieves up to a 4000× reduction in scale without losing attack-chain integrity.

Altinisik et al. proposed ProvG-Searcher [15], an efficient provenance graph search method based on graph representation learning. By formulating threat behavior search as a subgraph matching problem, it enables the detection of known APT behaviors in large-scale provenance graphs, providing an effective solution for simplified representations.

3.2. Feature Embedding and Representation Learning

Feature representation methods for APT provenance graphs can be broadly divided into traditional non-trainable approaches and deep learning-driven graph embedding methods. Traditional approaches are simple and lightweight, suitable for resource-constrained environments or scenarios requiring strong interpretability. In contrast, deep learning methods provide more powerful structural modeling and contextual expressiveness, offering advantages in capturing complex patterns and improving detection accuracy.

3.2.1. Traditional Methods

Perozzi et al. proposed DeepWalk [16], which extends language modeling techniques from natural language processing to graph analysis. By applying truncated random walks to capture local graph structures and treating the walks as “sentences”, DeepWalk employs the SkipGram model to learn the low-dimensional continuous embeddings of vertices, encoding network structural patterns. Building on this, Grover and Leskovec introduced node2vec [17], which incorporates a biased random walk strategy with tunable parameters. This enables the flexible exploration of graph neighborhoods to capture both homophily (structural similarity) and structural equivalence (role similarity). Node2vec effectively extracts complex relational patterns, demonstrates strong scalability, and achieves superior performance to DeepWalk in tasks such as multi-label classification and link prediction.

Han et al. proposed UNICORN [18], which leverages full-system runtime provenance graphs to capture causal relationships between system entities. It employs an improved Weisfeiler–Lehman subtree kernel to construct graph histograms that encode multi-hop structural and temporal features. Combined with HistoSketch, it produces fixed-size sketches for the efficient summarization of dynamic graphs. Furthermore, UNICORN builds temporal evolution models during training to characterize the progression of normal system behavior, thereby resisting poisoning during deployment. By comparing real-time sketches with learned models, it detects anomalies effectively. This approach specifically targets the “low-and-slow” nature of APTs, addressing some of the limitations of traditional

provenance analysis in capturing long-term causality, resisting poisoning, and scaling to large systems, and it has demonstrated strong effectiveness in the relevant scenarios.

3.2.2. Deep Learning-Based Methods

With the widespread success of GNNs in various graph modeling tasks, they have also demonstrated significant advantages in embedding APT provenance graphs. GNNs jointly capture structural and attribute information and possess strong local dependency modeling capabilities, making them a mainstream approach for APT detection in recent years.

Classic models include the Graph Convolutional Network (GCN) [19] and the Graph Attention Network (GAT) [20]. GCN propagates and aggregates node features through a normalized adjacency matrix, enabling the capture of multi-hop neighborhood information with only a few layers. GAT introduces a self-attention mechanism that assigns different weights to neighbors, overcoming the limitation of treating all adjacent nodes equally in traditional graph convolution. Hamilton, Ying, and Leskovec proposed GraphSAGE [21], which adopts an inductive embedding mechanism that samples and aggregates neighbor features to generate embeddings for unseen nodes. This approach performs well on evolving and large-scale graphs, enabling transferable embeddings without retraining. Li et al. introduced the Gated Graph Sequence Neural Network (GGS-NN) [22], which combines GNNs with GRUs to embed message passing into a recurrent sequence structure, producing node- or graph-level sequence representations suitable for capturing temporal dynamics and multi-step reasoning, with applications in program analysis and attack-chain reconstruction.

Building upon these foundations, many researchers have incorporated self-supervised learning and dynamic modeling techniques. Jia et al. proposed the MAGIC framework [23], which leverages masked graph representation learning for self-supervised APT detection. Without relying on attack data or prior knowledge, MAGIC employs a graph masking autoencoder to extract deep features, learning low-dimensional embeddings that capture entity context and multi-hop interaction patterns. It enables multi-granularity detection while maintaining low computational overhead.

Qiao et al. proposed the Slot method [24], which integrates graph reinforcement learning to guide semantic and structural embeddings of nodes, uncovering multi-layer hidden relations in APT provenance graphs. This significantly enhances robustness against noise and mimicry attacks, addressing the limitations of traditional methods in resisting adversarial camouflage and capturing deeper behavioral associations.

Bahar et al. introduced CONTINUUM [25], which employs a spatial-temporal graph neural network to capture the multi-stage and long-term evolution of APT attacks. It addresses the challenge of jointly modeling entity interactions and temporal dynamics, and it combines spatio-temporal learning with federated learning to specifically handle the “low-and-slow” multi-stage nature of APTs, while preserving privacy in large-scale deployments.

4. GNN-Based Feature Representation

This chapter focuses on the representation of provenance graph features for APT attacks within the context of the power industry. The APT attack process involves multi-source heterogeneous data such as host logs, network traffic, threat intelligence, and control commands. These data sources exhibit significant differences in structure, semantics, and temporal granularity, resulting in highly dispersed and heterogeneous features that are difficult to process and poorly adaptable. Moreover, APT attacks are characterized by long-term persistence, multi-stage evolution, and dynamic strategies. Without a unified modeling framework, it is challenging to achieve cross-source provenance correlation and

causal analysis or uncover the underlying causes of attacks, thereby hindering model transferability and generalization.

To address these challenges, we propose a unified and interpretable modeling framework that ensures both explainability and generalization. Built upon the W3C PROV-DM standard, the framework integrates multi-source data normalization, event-level provenance graph construction, and an autoencoder-based feature enhancement mechanism to achieve temporal correlation modeling of APT behaviors. The significance of this framework lies in the following aspects:

1. Unified representation and semantic fusion of multi-source data: Through consistent relationship modeling and temporal snapshot partitioning, heterogeneous data are mapped into a unified semantic space, enabling global correlation and fusion across multi-source events.
2. Integration of feature modeling and detection mechanisms: The unified framework explicitly defines causal dependencies and temporal relationships among events, allowing the model to capture latent causality and abnormal behavior patterns, thereby improving generalization and interpretability for unseen attacks.

On this basis, we further incorporate GNN to propose a feature enhancement method for provenance graphs, enabling the learning of both the structural dependencies and temporal evolution patterns of APT attack behaviors. The overall architecture is illustrated in Figure 1.

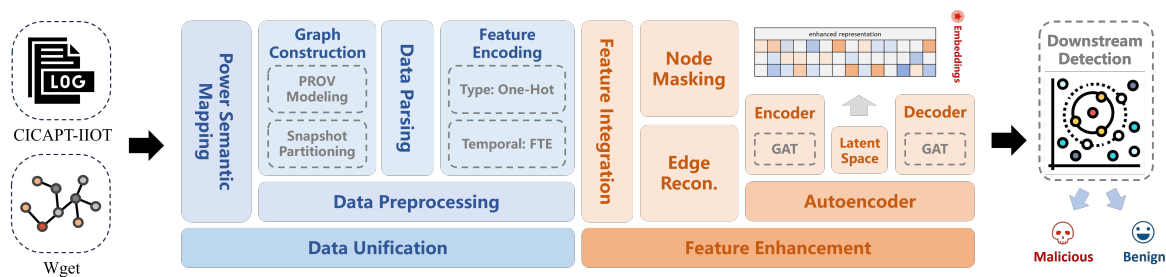


Figure 1. Architecture overview. The autoencoder adopts an “encode–decode–reconstruct” paradigm, performing self-supervised learning by minimizing the difference between the input and its reconstruction. Node masking hides part of the nodes, while edge reconstruction masks or fabricates certain edges to recover relationships and rebuild features using neighboring nodes and topological context. The latent space represents the enhanced high-dimensional feature embeddings.

Due to strict security and confidentiality requirements, power utilities rarely release APT-related datasets, and available data with power-specific contexts are extremely limited. To address this, we employ semantic mapping to transform generic APT data into provenance graphs enriched with power-domain semantics, thereby improving interpretability while preserving a general analytical framework.

At the data level, inconsistencies in recording formats and granularity across datasets often lead to incompatibility of features and structures. To resolve this, we propose the following unified graph construction method: entities and relations are mapped into the PROV model, and snapshots are generated based on timestamps. This converts heterogeneous, event-level data of varying granularity into standardized APT provenance graphs, while temporal encoding is applied to enhance the modeling of attack-chain dependencies.

At the model design level, we adopt the idea of a Masked Graph Autoencoder (MGAE) to build an encoder–decoder architecture. Node and edge features are encoded using a GAT, while a node-masking reconstruction mechanism is introduced to learn normal patterns. This design enables the model to generate more discriminative graph representations for anomaly detection and APT attack analysis.

We use a flowchart (see Figure 2) to clearly illustrate the specific operational process.

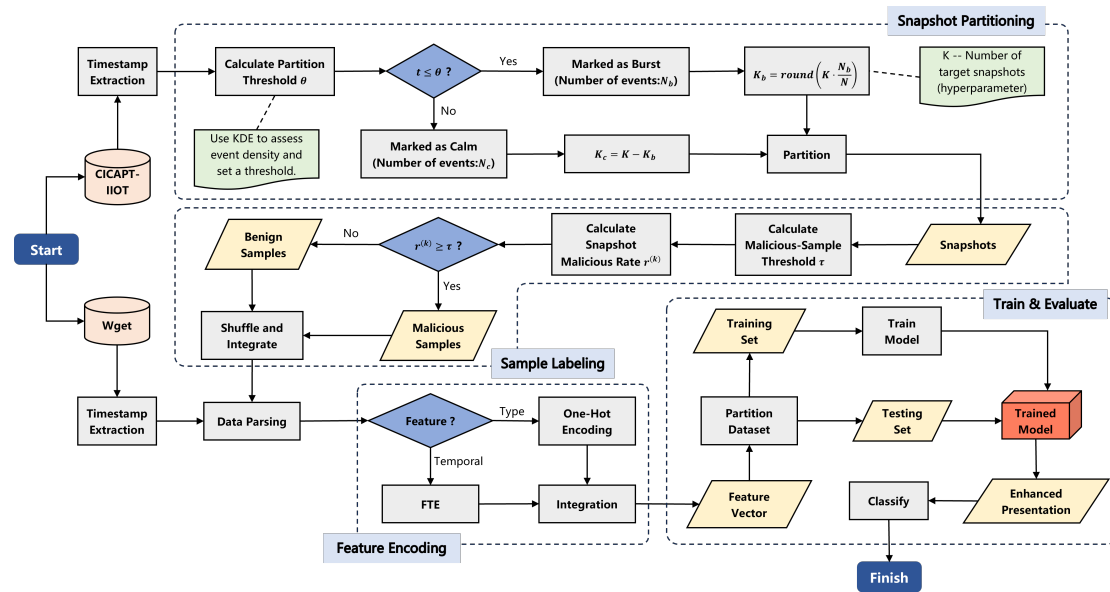


Figure 2. Overall workflow.

4.1. Power-Specific Semantic Mapping

4.1.1. W3C PROV-DM

The W3C Provenance Data Model (PROV-DM), proposed by the World Wide Web Consortium (W3C) in 2013, provides a domain-independent and extensible standardized framework for representing provenance information. It is designed to describe the generation and usage relationships among data, processes, and responsible agents, thereby enabling cross-system provenance data sharing and verification. The model is based on a Directed Acyclic Graph (DAG) structure, where activities precede the results they generate and derivations inherit from their predecessors. This naturally prevents cycles in provenance graphs and ensures that causal dependencies remain clear and traceable.

In PROV-DM, the nodes and edges of the provenance graph are described through type elements and semantic relations, with the complete definitions summarized in Table 1. By providing explicit conceptual definitions, PROV-DM abstracts events into a formalized graph structure that captures both temporal order and causal dependencies, as illustrated in Figure 3.

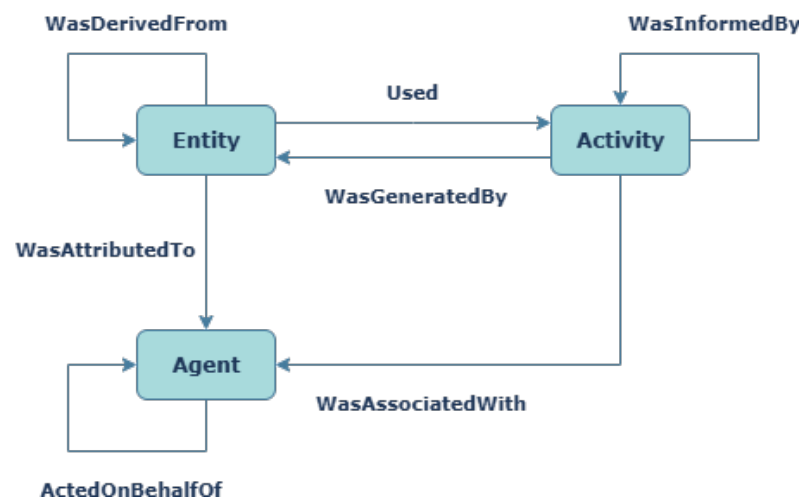


Figure 3. PROV core structures.

Table 1. Mapping of PROV core concepts.

PROV Concepts	Types or Relations	Name
Entity Activity Agent	PROV-DM Types	Entity Activity Agent
Generation Usage Communication Derivation Attribution Association Delegation	PROV-DM Relations	WasGeneratedBy Used WasInformedBy WasDerivedFrom WasAttributedTo WasAssociatedWith ActedOnBehalfOf

The raw data of APT attacks are often derived from system logs, captured by specialized tools such as CamFlow at the kernel level, which record interactions among processes, files, and network objects. These data can also be standardized under the PROV model through type mapping, as in the following examples:

- Entity: Data objects with a certain state or content, such as files, network packets, or sockets.
- Activity: Operations or processes, such as program execution or network downloads.
- Agent: Acting subjects, such as users or system services.

Representative relation mappings include the following:

- used: Read operations.
- wasGeneratedBy: Write operations.
- wasInformedBy: Inter-process communication.

Through such mappings, raw log data can be structured into standardized provenance graphs. This enables provenance information to be presented in a human-readable graphical form, while also supporting machine reasoning, querying, and cross-system correlation analysis and visualization.

In modeling APT attack chains, introducing PROV semantics allows diverse attack steps to be expressed under a unified causal framework, enhancing the standardization and clarity of provenance analysis. For instance, “Process A creates Process B” can be represented as Process B (Entity) being generated by the activity of Process A, while a network connection can be modeled as a communication activity that used a source IP entity and generated a destination IP entity. This mapping fully leverages the semantic consistency of PROV; every event in an APT chain corresponds to a standardized PROV relation, enabling logs from heterogeneous sources to be integrated into a unified semantic framework for correlation analysis. Moreover, as a domain-independent model, PROV explicitly supports the addition of domain-specific semantics. This extensibility strengthens the interpretability of APT attack chains in specific business contexts—for example, in the power industry, specialized agent types such as dispatching systems or monitoring devices can be incorporated.

4.1.2. Semantic Mapping

Most available APT data are derived from generic system logs, whereas the power system has unique business structures, critical asset types, and operational processes. Dedicated APT datasets for power systems remain extremely scarce. To make APT provenance graphs more interpretable and business-relevant in the power domain, it is necessary to integrate generic PROV semantics with domain-specific contexts, thereby constructing

semantic mappings that accurately characterize the potential impact of attack chains on power system operations.

Based on the analysis of technical documentation from power enterprises such as China Southern Power Grid, as well as case studies of APT attacks targeting power systems, we identified nine core concepts that reflect the operational logic of the power domain for node mapping. These include the dispatching control center, distribution automation system, monitoring terminal, backend server (dispatching master station), remote control, data acquisition, task scheduling, file transfer, and command execution. Together, these concepts cover both critical infrastructures in the power system and common operation types observed in APT campaigns, ensuring business representativeness as well as attack relevance.

Meanwhile, the generic relation types in PROV are preserved, but their semantics are interpreted according to node types and business contexts. In other words, each edge is expressed as a combination of “node type pair + relation type”, which specifies its meaning in the corresponding power system scenario.

To explicitly distinguish elements specific to the power domain, we define a unified namespace *ps* (power system) and extend the PROV standard accordingly. For each node, we establish a mapping between its PROV type and the corresponding power system concept, along with its definition and description, as summarized in Table 2.

Table 2. Namespace definitions for power system concepts.

Node Type	PROV Type	ps Namespace	Description
Dispatch Control Center	Agent	ps:DispatchControlCenter	Core institution for grid operation, responsible for monitoring, load dispatching, and fault handling.
Distribution Automation System	Entity	ps:DistributionAutomationSystem	Online monitoring and management of distribution networks, enabling automation, fault isolation, and recovery.
Monitoring Terminal	Entity	ps:MonitoringTerminal	Remote terminal installed in substations or field sites for data acquisition and remote control execution.
Dispatch Server	Entity	ps:DispatchServer	Server hosting the main dispatching functions, responsible for data aggregation, analysis, and command issuing.
Remote Control	Activity	ps:RemoteControlOperation	Operation process of remotely controlling generation or distribution equipment by dispatch centers or systems.
Data Acquisition	Activity	ps:DataAcquisition	Process of collecting operational parameters or status information from power equipment.
Task Dispatching	Activity	ps:TaskDispatching	Process of planning, optimizing, and allocating operational tasks in power systems.
File Transfer	Activity	ps:FileTransfer	Process of transferring files or data (configurations, logs, backups, etc.) across networks.
Command Execution	Activity	ps:CommandExecution	Process of executing commands or running programs on power system devices or servers.

Based on the Ukraine power grid cyberattack incident [26], we construct a simplified attack scenario to map the APT attack chain. In this scenario, the attacker uses malicious software to remotely control circuit breakers in a substation (within the power SCADA system), leading to large-scale regional outages. By applying the *ps* namespace, the semantic events can be extracted as follows:

- The attacker initiates a remote control operation on the circuit breaker: `ps:Attacker` wasAssociatedWith `ps:RemoteControlOperation`.
- The operation uses the dispatch master station as a pivot to execute control commands: `ps:RemoteControlOperation` used `ps:DispatchServer`.
- The operation triggers the deployment of a malicious module: `ps:RemoteControlOperation` wasGeneratedBy `MaliciousModule`.
- The operation targets and controls the distribution automation device (circuit breaker) to perform a forced trip: `ps:RemoteControlOperation` used `ps:DistributionAutomationSystem`.

Based on the above semantic modeling, the corresponding provenance graph can be constructed, as shown in Figure 4.

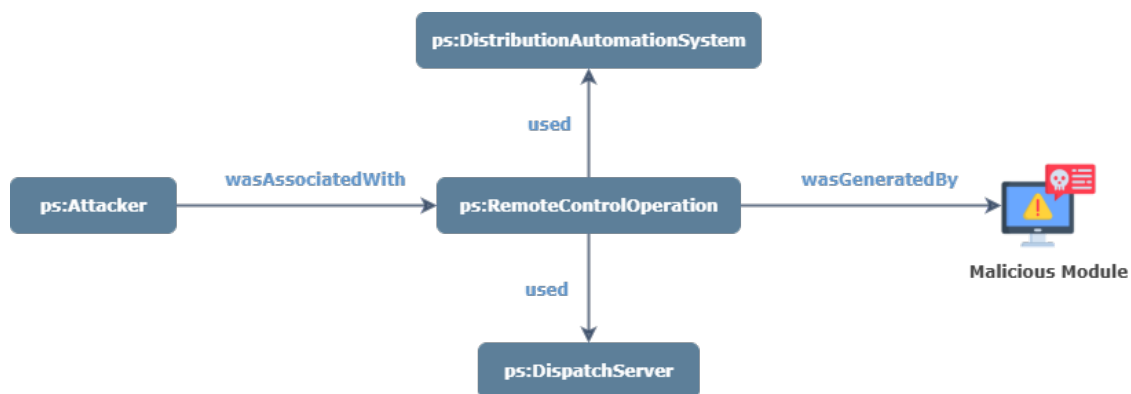


Figure 4. Power grid attack provenance graph.

From the analysis of the mapping graph, it can be observed that the attacker leveraged the dispatching system as a pivot, using remote control activities to manipulate distribution system devices and ultimately triggering a physical fault (power outage). Based on such relational mappings, it is possible not only to extract semantics from log events to construct provenance graphs, but also to derive the APT attack chain in reverse from the provenance graph, thereby enhancing interpretability in domain-specific contexts.

4.2. Data Preprocessing

4.2.1. Construction of APT Provenance Graphs

PROV Modeling

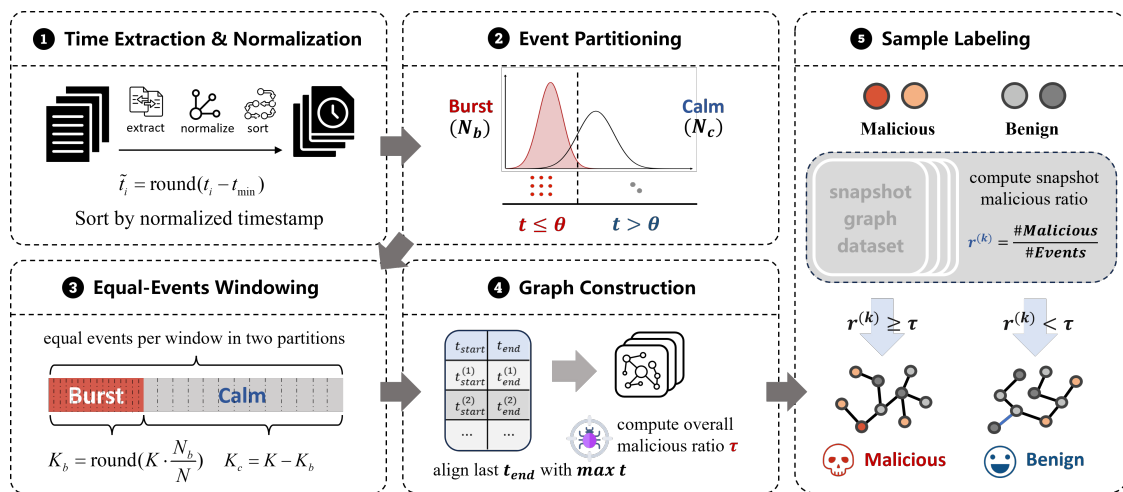
In heterogeneous multi-source security event data, different datasets often vary significantly in recording formats, field definitions, and granularity. To map such raw data into a unified provenance representation framework, we first perform semantic modeling based on the PROV standard. Taking the CICAPT-IIOT dataset [27] as an example, event nodes are categorized into two types as follows: Artifact and Process, where Artifact is further refined into four subtypes—Directory, File, Link, and Socket. Relations include the following four types: Used, WasGeneratedBy, WasTriggeredBy, and WasDerivedFrom. By analyzing event types and their interrelations in the raw data, we map both nodes and relations into the semantic space of PROV-DM, as summarized in Table 3. Subsequently, by applying the power system semantic mapping (Section 4.1), the data are contextualized with power-domain semantics, thereby achieving unified semantic modeling.

Table 3. CICAPT-IIOT to PROV-DM mapping.

Type Mapping		
CICAPT-IIOT	PROV-DM	Description
Artifact	prov:Entity	Representing identifiable resources, subtypes: Directory, File, Link, Socket.
Process	prov:Activity	Representing an executed behavior over a period of time.
Relation Mapping		
CICAPT-IIOT	Source → Target	PROV-DM Relation
Used	Process → Artifact	prov:used
WasGeneratedBy	Artifact → Process	prov:wasGeneratedBy
WasDerivedFrom	Artifact → Artifact	prov:wasDerivedFrom
WasTriggeredBy	Process → Process	prov:wasInformedBy

Snapshot Partitioning

Many security events are recorded at the granularity of individual events. To obtain sufficient provenance graph samples, consecutive events need to be aggregated in chronological order to construct snapshots that reflect the interactions between entities and activities within a specific time window. Since temporal features play a central role in event correlation and APT attack chain reconstruction, snapshot partitioning is essential. As an example, we illustrate the process of snapshot partitioning using a subset of the CICAPT-IoT dataset, as shown in Figure 5.

**Figure 5.** Snapshot partitioning.

- i. **Timestamp Extraction and Normalization:** Raw logs often contain multiple time-related fields (e.g., time, seen time, and start time). To ensure temporal consistency, we extract a single timestamp t_i based on a predefined priority as follows: if the primary field is missing, we fall back to the next available field; if no timestamp field exists, the record is discarded. All events are then chronologically ordered to generate a time sequence as follows:

$$\{(t_i, x_i, y_i)\}_{i=1}^N, \quad y_i \in \{0, 1\}$$

where x_i represents the event attributes and y_i the label, with 0 indicating benign samples and 1 indicating malicious samples.

Since timestamps are often collected from different sources, their recording precision and units may vary, and their values may span a wide range. To avoid the numerical instability caused by excessively large absolute time values, while still preserving the

relative temporal order and interval proportions, we normalize the timestamp fields as follows:

$$\tilde{t}_i = \text{round}(t_i - t_{\min})$$

- ii. Event Partitioning: Based on the normalized timeline $\tilde{t}$, we plot the event sequence and perform Kernel Density Estimation (KDE) on the two classes of $\tilde{t}$. As shown in Figure 6, malicious events exhibit significantly higher density in certain intervals, which we define as the burst region (Burst), while the remaining intervals are regarded as the calm region (Calm). Let the target number of snapshots be K and the burst threshold be θ . Events are divided into two segments according to $\tilde{t} \leq \theta$ and $\tilde{t} > \theta$, yielding event counts N_b and N_c , respectively. The number of snapshots in each region is then computed as follows:

$$K_b = \text{round}\left(K \cdot \frac{N_b}{N}\right), \quad K_c = K - K_b$$

Since events in the burst region are dense while those in the calm region are sparse, fixed-length time windows may result in oversized, over-saturated graphs in the burst region and undersized, overly sparse graphs in the calm region. To address this, we apply equal-event partitioning within each region so that every snapshot contains a comparable number of events and similar structural complexity, which facilitates stable model training and fair evaluation. Each index interval (s, e) is mapped back to a time range $[t_s, t_e - 1]$ to form the snapshot windows as follows:

$$\left\{ \left(t_{\text{start}}^{(k)}, t_{\text{end}}^{(k)}, n_{\text{events}}^{(k)} \right) \right\}_{k=1}^K$$

The right boundary of the last window is aligned with the global max t to ensure completeness. Within each window, events are further converted into a provenance graph, resulting in one graph sample per snapshot.

- iii. Sample Labeling: To prevent sparse malicious events from contaminating the entire snapshot and leading to an excessive number of malicious samples, we define a malicious threshold as $\tau = \frac{\sum_i 1[y_i=1]}{N}$. For the k -th snapshot, if its malicious ratio satisfies the following:

$$r^{(k)} = \frac{\sum_{i \in \text{win}_k} 1[y_i = 1]}{n_{\text{events}}^{(k)}} \geq \tau$$

it is labeled as a malicious snapshot; otherwise, it is labeled as benign. The threshold τ can be adjusted in combination with KDE results for sensitivity analysis.

Through this process, event-level data are transformed into graph-level samples, and the entire procedure is summarized in Algorithm 1.

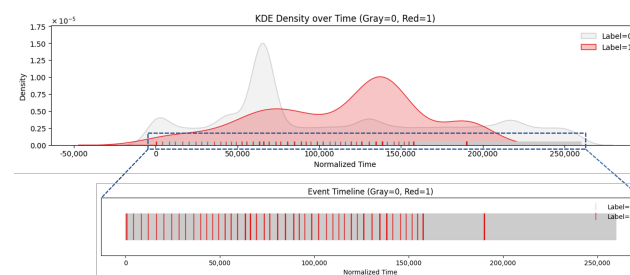


Figure 6. Sample distribution.

Algorithm 1: Snapshot division

Input: $\mathcal{L}$ (event log), K (target snapshots), h (KDE bandwidth)
Output: $\{(t_{\text{start}}[k], t_{\text{end}}[k], n[k], \text{label}[k])\}_{k=1}^K$

```

1 events  $\leftarrow$  ExtractTimestamps( $\mathcal{L}$ );
2 SortByTime(events);
3  $N \leftarrow \text{Length}(\text{events})$ ;  $t_{\min} \leftarrow \text{EarliestTime}$ ;  $t_{\max} \leftarrow \text{LatestTime}$ ;
4 normTime  $\leftarrow \text{Round}(\text{Time}(\text{events}) - t_{\min})$ ;
5  $\theta \leftarrow \text{EstimateBurstThreshold}(\text{normTime}, h)$ ;
6 burst_idx  $\leftarrow \{i \mid \text{normTime}[i] \leq \theta\}$ ;
7 calm_idx  $\leftarrow \{i \mid \text{normTime}[i] > \theta\}$ ;
8  $K_b \leftarrow 0$ ;  $K_c \leftarrow 0$ ;
9 for  $i \leftarrow 1$  to  $K$  do
10   if  $i \bmod 2 = 1$  then
11      $K_b \leftarrow K_b + 1$ ;
12     // Assign to Burst segment
13   else
14      $K_c \leftarrow K_c + 1$ ;
15     // Assign to Calm segment
16 wins  $\leftarrow \text{MergeAscending}(\text{EqualSizeChunks}(\text{burst\_idx}, K_b),$ 
17    $\text{EqualSizeChunks}(\text{calm\_idx}, K_c))$ ;
18  $\tau \leftarrow \text{GlobalMaliciousRate}(\text{events})$ ;
19 for  $k \leftarrow 1$  to  $K$  do
20   idx_list  $\leftarrow \text{wins}[k]$ ;
21    $n[k] \leftarrow \text{Length}(\text{idx\_list})$ ;
22    $t_{\text{start}}[k] \leftarrow t_{\min} + \min(\text{normTime}[\text{idx\_list}])$ ;
23    $t_{\text{end}}[k] \leftarrow t_{\min} + \max(\text{normTime}[\text{idx\_list}])$ ;
24    $r \leftarrow \text{MaliciousRate}(\text{events}[\text{idx\_list}])$ ;
25   if  $r \geq \tau$  then
26     label[k]  $\leftarrow$  malicious;
27   else
28     label[k]  $\leftarrow$  benign;
29  $t_{\text{end}}[K] \leftarrow t_{\max}$ ;
30 return  $\{(t_{\text{start}}[k], t_{\text{end}}[k], n[k], \text{label}[k])\}_{k=1}^K$ ;
```

4.2.2. Feature Vector Construction**Data Parsing**

The raw log data are first parsed to extract the node set $\mathcal{V}_{\text{raw}}$ along with the node types. To unify node identifiers, a hash function is applied to map the ID field into a unique and stable index as follows:

$$\phi_v(v) = \text{hash}(v), \quad v \in \mathcal{V}_{\text{raw}}$$

Next, the edge set $\mathcal{E}_{\text{raw}}$ is extracted together with their types as follows:

$$\mathcal{E}_{\text{raw}} = \{e \mid e = (\phi_v(v_s), \phi_v(v_t), r)\}$$

where r denotes the edge type. Finally, a directed graph $G = (\mathcal{V}, \mathcal{E})$ is constructed using networkx, and redundant edges are removed, i.e., if multiple edges of the same type exist between two nodes, they are merged into a single edge.

One-Hot Encoding

Since both node and edge types are categorical features, we apply one-hot encoding to transform them into vectorized representations for subsequent model training. Let the set of node types be

$$\mathcal{T}_v = \{\tau_1^v, \tau_2^v, \dots, \tau_{|\mathcal{T}_v|}^v\}$$

and the set of edge types be

$$\mathcal{T}_e = \{\tau_1^e, \tau_2^e, \dots, \tau_{|\mathcal{T}_e|}^e\}$$

For any node $v \in \mathcal{V}$, its type is denoted as $\text{type}(v) \in \mathcal{T}_v$, and its one-hot feature vector is defined as follows:

$$x_v \in \{0, 1\}^{|\mathcal{T}_v|}, \quad x_v[k] = \begin{cases} 1, & \text{type}(v) = \tau_k^v \\ 0, & \text{type}(v) \neq \tau_k^v \end{cases}$$

Similarly, for any edge $e \in \mathcal{E}$, its type is denoted as $\text{type}(e) \in \mathcal{T}_e$, and its one-hot feature vector is defined as follows:

$$x_e \in \{0, 1\}^{|\mathcal{T}_e|}, \quad x_e[k] = \begin{cases} 1, & \text{type}(e) = \tau_k^e \\ 0, & \text{type}(e) \neq \tau_k^e \end{cases}$$

Here, $|\mathcal{T}_v|$ and $|\mathcal{T}_e|$ denote the number of node and edge types, which also determine the dimensionality of their one-hot feature vectors.

4.2.3. Temporal Feature Encoding

To better utilize temporal information, we adopt the Functional Time Encoding (FTE) method [28] to encode timestamps.

FTE views the mapping from time to vector space as a continuous functional transformation as follows:

$$\Phi : \mathcal{T} \rightarrow \mathbb{R}^{d_\tau}$$

It focuses on relative time differences rather than absolute timestamps, and it expresses temporal similarity as a translation-invariant kernel. Thus, it learns patterns over time intervals $|t_2 - t_1|$ and facilitates the discovery of periodic behaviors as follows:

$$K(t_1, t_2) = \langle \Phi(t_1), \Phi(t_2) \rangle = \psi(t_1 - t_2)$$

Since the kernel is continuous and Positive Semi-Definite (PSD), by Bochner's theorem, it can be represented as the Fourier transform of a probability measure. This yields an expected form on cosine basis functions, which can be approximated in finite dimensions via Random Fourier Features (RFFs) as follows:

$$\text{FTE}_d(t) = \sqrt{\frac{1}{d}} [\cos(\omega_1 t + \phi_1), \dots, \cos(\omega_d t + \phi_d)]$$

where ω_i are trainable frequency parameters representing the i -th basis frequency, ϕ_i are trainable phase shifts, and d is the encoding dimension controlling the number of frequency components extracted. In practice, we set $d_e = 8$ so that the temporal features are comparable in weight to type features. Finally, the type feature vectors and temporal feature vectors are concatenated to form the final feature vector.

The overall procedure is summarized in Algorithm 2.

Algorithm 2: Graph feature construction

Input: $\mathcal{L}$ (raw log), d_v (node time dim), d_e (edge time dim)
Output: $G = (\mathcal{V}, \mathcal{E})$ with feature vectors

- 1 $\mathcal{V}_{raw} \leftarrow \text{ExtractNodes}(\mathcal{L});$
- 2 Assign unique IDs: $\phi_v(v) \leftarrow \text{Hash}(v)$ for $v \in \mathcal{V}_{raw};$
- 3 $\mathcal{E}_{raw} \leftarrow \text{ExtractEdges}(\mathcal{L}, \phi_v);$
- 4 Remove redundant edges with same type between same nodes;
- 5 $G \leftarrow \text{BuildDirectedGraph}(\mathcal{V}, \mathcal{E});$
- 6 $\mathcal{T}_v \leftarrow \text{UniqueNodeTypes}(\mathcal{V});$
- 7 $\mathcal{T}_e \leftarrow \text{UniqueEdgeTypes}(\mathcal{E});$
- 8 **foreach** $v \in \mathcal{V}$ **do**
- 9 $x_v \leftarrow \text{ZeroVector}(|\mathcal{T}_v|);$
- 10 $x_v[k] \leftarrow 1$ if $\text{Type}(v) = \tau_k^v;$
- 11 $f_v \leftarrow x_v;$
- 12 **foreach** $e \in \mathcal{E}$ **do**
- 13 $x_e \leftarrow \text{ZeroVector}(|\mathcal{T}_e|);$
- 14 $x_e[k] \leftarrow 1$ if $\text{Type}(e) = \tau_k^e;$
- 15 Init $\{\omega'_i, \phi'_i\}_{i=1}^{d_e}$ as trainable params;
- 16 **foreach** $e \in \mathcal{E}$ **do**
- 17 $t \leftarrow \text{Timestamp}(e);$
- 18 $t_e \leftarrow \sqrt{\frac{1}{d_e}} [\cos(\omega'_1 t + \phi'_1), \dots, \cos(\omega'_{d_e} t + \phi'_{d_e})];$
- 19 $f_e \leftarrow \text{Concat}(x_e, t_e);$
- 20 **return** G with $\{f_v\}$ and $\{f_e\};$

4.3. Model Design

In APT detection, malicious samples are typically much rarer than benign ones. An autoencoder, through its encoding–decoding–reconstruction mechanism, minimizes the discrepancy between input and reconstruction, enabling self-supervised learning that fully leverages abundant benign samples even in the absence of labels. Compared with conventional GNN-based approaches, this provides significant advantages in APT provenance graph detection. Building on this, we introduce an autoencoder architecture inspired by the mask-based reconstruction strategy of MAGIC [23]. Specifically, a subset of nodes is masked, and their original features are reconstructed using neighborhood information and global topology. The reconstruction error is used as the optimization objective, allowing the model to capture the implicit normal patterns embedded in benign samples. The encoder maps data into the latent space, where feature representations are enhanced.

4.3.1. Architecture

After the preprocessing described in Section 4.2, we obtain standardized provenance graph data. To align the dimensionality and normalize the scale of heterogeneous features from different sources, thereby improving model scalability, we introduce a linear transformation layer at the input stage. This facilitates integration while preserving both the temporal dynamics of APT behaviors and the semantic information of event types.

Subsequently, benign samples are selected, and a random subset of node features is masked. The masked features are replaced by learnable noise vectors, which, together with the unmasked nodes, are fed into the encoder. The encoder generates latent node representations within the structural context, which are then linearly transformed and passed to the decoder. The decoder reconstructs the masked node attributes on the original graph topology.

In addition, to ensure that the latent representations explicitly encode structural patterns, we incorporate a structural reconstruction branch as follows: edges are classified as positive or negative in a binary prediction task. The model jointly minimizes the node reconstruction loss and the edge reconstruction loss, thereby aligning representations with both attribute consistency and topological consistency. The overall model architecture is illustrated in Figure 7.

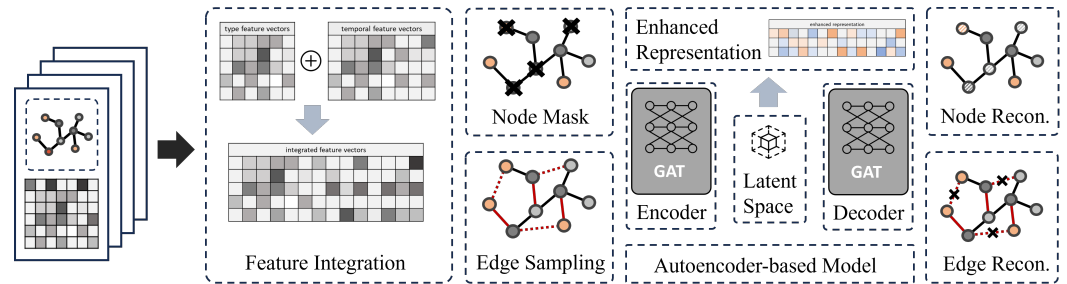


Figure 7. System architecture. Colored nodes indicate malicious events, gray nodes denote benign events. In Node Mask, “x” marks masked nodes; in Node Recon., dashed nodes are reconstructed. In Edge Sampling, solid and dashed red lines indicate positive and negative samples, respectively; in Edge Recon., solid red lines are reconstructed positive edges, and dashed “x” edges are removed negative edges.

Through training, the model is forced to infer missing node features from neighborhood and global structures, thereby effectively learning the co-occurrence patterns of structure and attributes in benign samples. In the spatial dimension, the encoder employs spatial convolution-based GNNs such as the GAT, where multi-head attention dynamically models the association strengths between nodes and their neighbors, capturing anomalous behavioral features reflected in local connectivity patterns of APT attack chains. In the temporal dimension, the model integrates functional time encoding with structural learning, explicitly embedding the temporal dependencies of event sequences into node representations. This enables the system to recognize stealthy attacks that unfold progressively across multiple stages.

With this design, the feature enhancement stage produces high-quality embeddings that simultaneously encode spatial locality, global topology, and temporal dynamics. These representations provide a stable and discriminative foundation for downstream detectors such as KNN.

4.3.2. Mask-Based Reconstruction

In the node reconstruction stage, let the input provenance graph be $G = (V, E)$ with $|V| = N$ nodes and feature matrix $X \in \mathbb{R}^{N \times d}$. Given a masking ratio ρ , we first generate a random permutation of node indices π , and we select the first $M = \lfloor \rho N \rfloor$ nodes as the masked set $\mathcal{M}$, while the remaining nodes form the unmasked set $\mathcal{K}$. To eliminate explicit masking indicators and make the training input distribution closer to real data, the features of masked nodes are replaced with random noise vectors sampled according to the statistics of the current batch. Specifically, the mean and standard deviation are computed from the unmasked features $x_i \mid i \in \mathcal{K}$ as follows:

$$\mu = \frac{1}{|\mathcal{K}|} \sum_{i \in \mathcal{K}} x_i, \quad \sigma = \sqrt{\frac{1}{|\mathcal{K}|} \sum_{i \in \mathcal{K}} (x_i - \mu)^2 + \varepsilon}$$

Noise vectors are then sampled from a Gaussian distribution as follows:

$$\tilde{x}_{\mathcal{M}} \sim \mathcal{N}(\mu, \lambda^2 \sigma^2)$$

where λ is a noise intensity coefficient controlling the similarity between noise and real features. The masked feature matrix X' is then fed into the encoder for representation learning. Introducing random noise masking serves multiple purposes, outlined as follows: (i) it removes the “detectability” of fixed mask tokens, preventing the model from trivially identifying masked nodes without inference; (ii) it forces the encoder to reconstruct masked nodes using graph structure and contextual information, thereby capturing the co-occurrence between node attributes and structural patterns more effectively; and (iii) since the noise distribution is aligned with the real feature distribution, feature transfer between masked and unmasked nodes becomes smoother, improving training stability and generalization.

In the edge reconstruction stage, the model concatenates multi-layer node representations obtained from the encoder. A certain number of positive samples are randomly drawn from the real edge set, while an equal number of negative samples are generated through global uniform negative sampling. The model then predicts link probabilities for the sampled edge pairs.

4.3.3. Encoder

The encoder employs a multi-layer Graph Attention Network (GAT), which performs adaptive weighted aggregation of node neighborhood information under the message passing paradigm. In this way, the raw features—fused from type and time information—are mapped into more discriminative latent embeddings, serving as the foundation for downstream feature enhancement. For a layer ℓ , the message from a neighbor $v \rightarrow u$ is defined as follows:

$$m_{u \leftarrow v}^{(\ell)} = \alpha_{uv}^{(\ell)} W^{(\ell)} h_v^{(\ell)},$$

and aggregated as follows:

$$\tilde{h}_u^{(\ell)} = \sum_{v \in \mathcal{N}(u)} m_{u \leftarrow v}^{(\ell)},$$

followed by the update to obtain the next-layer representation, expressed as follows:

$$h_u^{(\ell+1)} = \sigma(\tilde{h}_u^{(\ell)}).$$

The attention weights $\alpha_{uv}^{(\ell)}$ are computed via self-attention scoring as follows:

$$e_{uv}^{(\ell)} = \text{LeakyReLU}\left(a^\top \left[W^{(\ell)} h_u^{(\ell)} \parallel W^{(\ell)} h_v^{(\ell)} \right]\right)$$

$$\alpha_{uv}^{(\ell)} = \frac{\exp(e_{uv}^{(\ell)})}{\sum_{w \in \mathcal{N}(u)} \exp(e_{uw}^{(\ell)})}$$

thus enabling the learnable, differentiated weighting of neighbors.

To capture information from different receptive fields, the encoder retains hidden representations from all layers and concatenates them in the feature dimension to form multi-scale representations as follows:

$$z_u = \left[h_u^{(1)} \parallel h_u^{(2)} \parallel \dots \parallel h_u^{(L)} \right],$$

Since different layers correspond to neighborhoods of varying ranges and structural scales, cross-layer fusion significantly improves robustness and separability in complex graph structures, capturing both local and long-range dependencies.

The set z_u represents the embeddings of raw samples mapped into the latent space. Compared with handcrafted features, these embeddings integrate local–global structural

relations and temporal semantics through attention-weighted message passing, and they can therefore be regarded as enhanced feature representations. On the one hand, they provide informative and noise-controlled inputs for the decoder's Mask-based Reconstruction (node-side SCE) and structural reconstruction (edge-side BCE); on the other hand, they supply more discriminative distance metrics for downstream unsupervised detection models.

4.3.4. Decoder

The decoder reconstructs the attributes of masked nodes on the original topology using the latent representations produced by the encoder, thereby projecting the structural context + latent semantics back into the aligned feature space. In implementation, we employ a shallow GAT-based decoder, preceded by a linear bridging layer that compresses the concatenated multi-layer hidden states from the encoder into a dimension consistent with the decoder input. The compressed representation is then used as the initial decoding feature of each node. This shallow decoder design follows the principles of graph mask-based autoencoders as follows: deep encoding with shallow decoding encourages information to be carried in the latent space while preventing the decoder from being overly powerful, which could otherwise “memorize” the input distribution and harm generalization.

Let z_u denote the latent representation of node u after linear bridging. Under the adjacency constraints of graph G , the decoder performs attention-based message passing as follows: neighbor information is aggregated with GAT's self-attention weights α_{uv} , and the node state is updated to yield the reconstructed vector $\hat{x}_u \in \mathbb{R}^n$. This paradigm allows the decoder to explicitly leverage neighborhood context to recover masked node features, rather than relying solely on the node's own latent embedding. Edge-level attention scoring and softmax normalization ensure that neighbors are assigned different levels of importance, which is particularly critical for heterogeneous and non-uniform provenance graphs.

4.3.5. Loss Function

The overall optimization objective consists of the following two components: node reconstruction loss and edge reconstruction loss, which constrain attribute consistency and structural consistency, respectively.

For node reconstruction, we adopt the Scaled Cosine Error (SCE), applied only to the masked node set. SCE measures the directional similarity between reconstructed features $\hat{x}_i$ and original features x_i via cosine similarity, while a scaling factor α is used to adjust for magnitude differences. This emphasizes semantic similarity in the embedding space while de-emphasizing absolute numerical discrepancies. Such a metric encourages the model to preserve the semantic patterns of nodes during reconstruction, stabilizing the latent representations in directional space and improving discriminability.

For edge reconstruction, we use Binary Cross-Entropy (BCE). During training, positive edges are randomly sampled from the true edge set, while an equal number of negative edges are generated through global uniform negative sampling. For each node pair (z_u, z_v) , a two-layer perceptron predicts the link probability p_{uv} . The predicted values are compared with ground-truth labels (1 for positive edges, 0 for negative edges) to compute the BCE loss. This explicitly enforces the topological consistency of latent representations, enabling the model to not only reconstruct node attributes but also accurately capture connectivity patterns.

The final objective is given by the following:

$$\mathcal{L} = \mathcal{L}_{\text{node}}^{(\text{SCE})} + \mathcal{L}_{\text{edge}}^{(\text{BCE})}$$

This joint optimization strategy ensures that both the semantic information and structural dependencies of APT provenance graphs are preserved during learning. As a result, the embeddings generated by the encoder remain highly sensitive and discriminative when facing anomalies in node attributes or structural perturbations, thereby providing a solid foundation for downstream feature enhancement and APT attack detection.

4.3.6. Downstream Detection

In the downstream stage, the trained model is used as a feature extractor. Specifically, we obtain node-level latent embeddings $\{z_u\}_{u \in V}$ through the encoder interface. Since these embeddings are node-level, whereas downstream detection requires fixed-length graph-level vectors, we apply a pooling operator to aggregate node embeddings into a graph-level representation of h_G as follows:

$$h_G = \text{POOL}(\{z_u \mid u \in V\}),$$

which balances the contributions of different node types to the overall representation. The resulting h_G serves as the enhanced graph-level feature; it originates from the encoder's latent mapping and integrates node attributes with neighborhood structure into a compact, more discriminative representation.

The features and labels are collected per graph and concatenated into the evaluation matrix X and label vector y . To avoid data leakage, we use a separate evaluation dataset different from the training and self-supervised learning stages, and the encoder is employed solely as a frozen feature extractor. For downstream detection, we adopt K-Nearest Neighbors (KNN) as the classifier.

KNN measures sample similarity in the latent space using cosine distance and determines class labels according to the distribution of neighbors. As a non-parametric method, KNN does not rely on an additional training process, directly reflecting the discriminability of latent features and avoiding overfitting risks associated with complex models. In the context of APT detection—where attack behaviors are often stealthy and multi-stage, and data scarcity with class imbalance is common—the non-parametric nature of KNN provides greater robustness, while highlighting the separability gains achieved through representation learning.

Overall, the pooled h_G remains sensitive to both global structure and local anomalies. KNN-based evaluation offers a simple and objective measure of the encoder's feature discriminability, validating the effectiveness and stability of enhanced representations, while also serving as a baseline reference for the design of more complex detection models.

4.3.7. Attack Detection

To avoid reliance on predefined attack templates, this study adopts a whitelist-based approach combining benign pattern learning with deviation detection. During training, only benign samples are used, and an autoencoder is employed to model the joint structural-temporal-semantic distribution of provenance graphs. The encoder maps each sample into a latent space that captures the temporal consistency, structural constraints, and semantic co-occurrence patterns of normal operational flows. After convergence, this latent space approximates the feature distribution of benign samples, forming a high-density region representing normal behaviors.

During inference, any input sample is projected into the same latent space and compared against the benign distribution. Samples whose latent representations align closely with benign patterns are deemed normal. In contrast, unknown attacks typically violate the temporal coherence, causal-chain structure, and PROV-based semantic constraints of benign data. Even if such attacks have not appeared in the training set, their embeddings

rarely fall within the high-density manifold of normal samples and are thus detected as deviations. This mechanism is inherently attack-agnostic; it identifies “normality”, classifying anything outside that norm as anomalous—without requiring prior knowledge of attack types or signatures.

5. Experiments and Evaluation

This section evaluates the effectiveness and applicability of the proposed method in the context of power systems. We conduct experiments on two public datasets, CICAPT-IIoT and Unicorn Wget, to assess the contributions of individual modules to overall performance. Furthermore, we compare our approach with baseline methods to summarize its advantages, limitations, and potential value for real-world power system cybersecurity.

5.1. Implementation Details

All experiments are conducted on a workstation equipped with an NVIDIA RTX 4090 GPU (48 GB memory) and CUDA 12.8, fully leveraging GPU parallelism to accelerate both training and inference. The software environment is based on Python 3.9, with PyTorch 2.2.2 (cu121) and DGL 1.1.3 (cu121). PyTorch, as a mainstream deep learning library, provides flexible tensor operations and dynamic computation graphs, while DGL (Deep Graph Library) offers efficient implementations specifically optimized for GNNs, ensuring stability and performance in graph computations.

For the model design, the encoder consists of four GAT-based layers that extract high-order structural and attribute information from provenance graphs. Each layer has a hidden feature dimension of 256, which provides sufficient capacity to capture complex graph patterns. The decoder is implemented as a single-layer structure, responsible for mapping the encoder’s latent representations back into the output space for reconstruction or classification.

During training, we employ the widely used Adam optimizer, which combines the advantages of AdaGrad and RMSProp to adaptively adjust learning rates, thereby accelerating convergence and improving stability. To prevent overfitting, we apply an early stopping strategy. For downstream detection, we use a KNN classifier with cosine distance as the metric. In particular, 70% of benign samples are used as the KNN reference set, and the number of neighbors K is set to 50% of the reference set size. This setup allows us to validate the effectiveness of the proposed feature representations in distinguishing between benign and malicious samples.

5.2. Datasets

We evaluate our method on two public provenance-based datasets, each applied in independent experimental settings. Table 4 summarizes their key characteristics. The two datasets differ significantly in source and scale, allowing us to comprehensively assess adaptability while maintaining scenario independence.

CICAPT-IIoT (Canadian Institute for Cybersecurity APT Datasets for IIoT Environment), proposed by Ghiasvand et al. [27], is a semi-synthetic provenance-based dataset specifically designed for APT detection in Industrial Internet of Things (IIoT) environments. It addresses the critical lack of IIoT-specific APT datasets in current security research. Built on the Brown IIoTbed framework, which combines virtual components with real devices, the dataset approximates the complexity and dynamic behavior of real IIoT systems. It provides a complete description of dependencies between entities and events, with provenance graphs capturing the evolution paths of APT attacks in IIoT systems.

Table 4. Dataset statistics.

Dataset	Granularity	Scenario	#Graphs	Avg.#Nodes	Avg.#Edges
CICAPT-IIoT	Event-level	benign	–	99,727	318,726
		malicious	–	332	70
Unicorn Wget	Graph-level	benign	125	39,445	153,621
		malicious	25	28,589	126,826

CICAPT-IIoT is built on a hybrid simulation and real-device platform, incorporating Raspberry Pi, OpenPLC, Modbus, sensors, and other components. It reflects the characteristics of smart grids with widely deployed IoT devices, high real-time requirements, and complex network connectivity, closely resembling the physical architecture and communication workflows of power systems. This IIoT architecture is particularly reusable in power scenarios such as smart substations and automated equipment control. Moreover, the dataset simulates more than 20 commonly used APT techniques, covering multiple critical stages of the APT lifecycle, and it is highly consistent with advanced persistent threats faced by power systems.

The Unicorn Wget dataset, on the other hand, is generated from CamFlow audit logs. It contains a total of 150 batch-level provenance graphs, including 125 benign and 25 malicious samples. The malicious samples simulate covert supply chain attacks designed by the UNICORN framework, where attacks are constructed through accesses to benign and malicious URLs. Compared with CICAPT-IIoT, the Unicorn Wget dataset is larger in scale and higher in complexity, making it suitable for evaluating detection methods under large-scale, complex log environments.

Since CICAPT-IIoT only provides raw records for provenance graph generation, we first apply the snapshot partitioning method (Section 4.2) to construct snapshot-level samples and annotate each sample. This yields a total of 49 benign samples and 15 malicious samples, as summarized in Table 5.

Table 5. Snapshot statistics for different scenarios.

File	Scenario	#Snapshots	Statistic	#Events	Duration	#Malicious Events	Malicious Ratio
Phase1_Provenance	Benign	34	Max	5448	12,301	–	–
			Min	5447	371	–	–
			Mean	5448	10,163	–	–
Phase2_Provenance	Benign	15	Max	5679	12,522	4	0
			Min	5449	6	0	0
			Mean	5571	6269	1	0
	Malicious	15	Max	5679	13,648	86	1.5%
			Min	5449	2387	6	0.1%
			Mean	5648	11,033	18	0.3%

To ensure training stability and evaluation reliability, the dataset was partitioned according to sample characteristics and model learning objectives. Since the proposed mask-based reconstruction model is designed to identify anomalies by learning normal behavioral patterns from benign samples, the training phase primarily relies on benign data for pattern modeling, while malicious samples are reserved for deviation detection and performance evaluation. Accordingly, 80% of the benign samples are used for training, and the remaining 20%, together with all malicious samples, form the test set. This ratio was determined through extensive experiments as the optimal configuration, achieving a balance among convergence speed, detection accuracy, and computational cost.

Specifically, when the training ratio falls below 80%, the model fails to adequately cover benign behavior patterns, leading to the overfitting of local modes and misclassification of boundary samples. When the ratio exceeds 80%, performance gains become marginal while training time and resource consumption increase substantially, and the reduced number of test samples weakens statistical reliability. The current configuration allows the model to fully converge and stably learn benign behavior while preserving sufficient test data to assess generalization under unseen scenarios.

Moreover, given the limited scale of the CICAPT-IIoT dataset and the fact that snapshot samples are manually constructed and labeled, excessively enlarging the training set would reduce test sample diversity and compromise the objectivity of evaluation. Overall, the adopted data partitioning achieves a sound balance among model convergence, detection performance, and evaluation validity.

5.3. Metrics

In this study, we adopt a set of widely used binary classification metrics to evaluate model performance. These metrics not only measure the overall discriminative capability of the model but also assess its accuracy and completeness in identifying positive and negative samples. Specifically, we use ROC AUC, Precision, Recall, F1-score, and the components of the confusion matrix (TP, TN, FP, and FN). The detailed formulas for these metrics are provided in Appendix A. Based on these metrics, we conduct a comprehensive evaluation of the proposed method on APT provenance detection tasks.

5.4. Ablation Study

To verify the contribution of each key module in the proposed framework to the overall performance, we design and conduct a series of ablation experiments. Under the same experimental workflow and data conditions, we selectively remove or replace certain components of the framework to analyze their impact on model performance, thereby validating their practical value in capturing APT attack patterns, enhancing robustness, and improving generalization.

Specifically, we construct three variant models based on the full framework by removing temporal encoding, node masking, or edge reconstruction, and we then evaluate the feature quality through downstream detection using a classifier.

In addition, we include the state-of-the-art model MAGIC [23] for comparison, which is also an autoencoder-based framework but does not incorporate temporal encoding mechanisms. This inclusion allows us to further assess the advantages brought by our temporal modeling and reconstruction modules.

- **Experiment 1: Without Temporal Encoding**
In this setting, the temporal encoding module is removed, and only static node and edge features are retained as input. Event timestamps are not embedded, allowing us to analyze the contribution of temporal encoding in capturing sequential dependencies within attack chains and improving temporal pattern modeling.
- **Experiment 2: Without Node Masking**
Here, the node masking mechanism is removed, meaning that during training, node features are no longer randomly masked and reconstructed. This experiment evaluates the role of node masking in modeling normal patterns and improving the robustness and generalization of feature representations.
- **Experiment 3: Without Edge Reconstruction**
In this experiment, the edge reconstruction task is removed, and the model only performs node feature prediction without attempting to reconstruct edge connections.

This setting validates the importance of structural reconstruction in enhancing the model’s ability to capture graph topology.

- Experiment 4: MAGIC

We include the state-of-the-art autoencoder-based model MAGIC for comparison. Unlike our framework, MAGIC does not model temporal features and employs a different node masking strategy. This comparison helps verify the effectiveness of our temporal encoding and mask-based reconstruction modules.

- Experiment 5: Full Model

This configuration retains all three modules—temporal encoding, node masking, and edge reconstruction—forming the complete generative self-supervised framework proposed in this study. It serves as the performance upper bound and is compared against the ablated variants to evaluate the overall effect of multi-module collaboration.

5.5. Results Analysis

Table 6 reports the performance of different experimental settings on the test sets. Overall, the model achieves relatively stable detection performance across both datasets.

Table 6. Ablation results on the Wget and CICAPT-IIOT datasets.

Dataset	Ablation Models	AUC	F1-Score	Precision	Recall	Confusion Matrix			
						TN	FN	TP	FP
Wget	w/o Temporal Encoding	0.95 (0.013)	0.93 (0.011)	0.95 (0.039)	0.92 (0.024)	14.00 (0.816)	1.67 (0.471)	18.33 (0.471)	1.00 (0.816)
	w/o Node Mask	0.97 (0.017)	0.95 (0.001)	0.92 (0.016)	0.99 (0.020)	13.20 (0.400)	0.20 (0.400)	19.80 (0.400)	1.80 (0.400)
	w/o Edge Recon.	0.95 (0.042)	0.93 (0.040)	0.91 (0.089)	0.96 (0.037)	12.80 (2.482)	0.80 (0.748)	19.20 (0.748)	2.20 (2.482)
	MAGIC	0.97 (0.015)	0.94 (0.015)	0.94 (0.034)	0.94 (0.058)	15.80 (0.748)	1.20 (1.166)	18.80 (1.166)	1.20 (0.748)
	Full Model	0.99 (0.003)	0.97 (0.013)	0.95 (0.029)	0.98 (0.040)	14.00 (0.632)	0.40 (0.800)	19.60 (0.800)	1.00 (0.632)
CICAPT-IIOT	w/o Temporal Encoding	0.88 (0.059)	0.88 (0.042)	0.88 (0.063)	0.90 (0.082)	3.20 (0.400)	0.60 (0.490)	5.40 (0.490)	0.80 (0.400)
	w/o Node Mask	0.87 (0.089)	0.90 (0.071)	0.84 (0.106)	0.97 (0.067)	2.80 (0.980)	0.20 (0.400)	5.80 (0.400)	1.20 (0.980)
	w/o Edge Recon.	0.86 (0.062)	0.87 (0.041)	0.83 (0.040)	0.93 (0.082)	2.80 (0.400)	0.40 (0.490)	5.60 (0.490)	1.20 (0.400)
	MAGIC	0.87 (0.112)	0.91 (0.035)	0.87 (0.105)	0.96 (0.049)	1.40 (1.356)	0.40 (0.490)	9.60 (0.490)	1.60 (1.356)
	Full Model	0.88 (0.083)	0.91 (0.012)	0.87 (0.051)	0.97 (0.035)	1.80 (0.980)	0.40 (0.490)	13.60 (0.490)	2.20 (0.980)

On the Wget dataset, the full model maintains high performance on all metrics, indicating that the proposed method effectively leverages temporal and structural information to model potential threats, achieving both stable and accurate identification of attack and benign samples.

In contrast, the CICAPT-IIoT dataset shows significantly lower scores. This performance gap is mainly due to the dataset’s limited scale and feature complexity. First, CICAPT-IIoT contains far fewer events than Wget, resulting in fewer snapshots and insufficient temporal dependencies for training. Second, each snapshot graph is relatively small, with limited node and edge diversity, restricting the richness of structural patterns. In addition, the distributional difference between benign and malicious samples in CICAPT-IIoT is less pronounced, further increasing the difficulty of distinguishing the two classes. Moreover, its graph-level samples are artificially constructed from raw event records rather than naturally formed, which may disrupt sample integrity and continuity, causing loss of contextual relationships and reducing fidelity in reflecting the evolution of attack chains. Nevertheless, despite lower absolute values, the model still maintains satisfactory detection capability, suggesting robustness even under limited data conditions.

These results demonstrate that the model effectively captures both temporal dependencies and structural features, while the feature enhancement mechanisms improve representational power, enabling reliable attack detection even in complex provenance graphs. Moreover, the relatively controlled performance fluctuations across settings re-

flect the adaptability of the model to diverse input features. However, its limited generalization on small-scale datasets highlights the need for further improvement under data-scarce scenarios.

5.5.1. Impact of Key Modules

On top of validating overall performance, we further analyze the role of individual modules through ablation experiments. Removing different modules reveals their specific contributions within the framework and highlights potential performance bottlenecks in their absence, thereby illustrating the practical value of temporal modeling, node masking, and structural enhancement for APT detection.

Experiment 1 (Without Temporal Encoding)

When temporal features are removed, the model shows performance degradation on both datasets. On the Wget dataset, AUC drops from 0.99 to 0.95 and F1-score from 0.97 to 0.93. Precision remains relatively stable, which can be regarded as normal fluctuation, while Recall decreases sharply from 0.98 to 0.92, directly lowering the F1-score. This indicates that without temporal encoding the model is more prone to miss anomalous events, failing to capture the continuity of attack behaviors. Although Precision remains high, it implies that the model labels fewer samples as “anomalous”, thereby reducing overall coverage.

A similar phenomenon is observed on the CICAPT-IIoT dataset. After removing temporal features, AUC and F1-score decrease to 0.88 and 0.88, while Recall drops significantly from 0.97 to 0.9. Given the dataset’s limited scale and small number of snapshots, the model already struggles to learn sufficient temporal dependencies. Without temporal encoding, the evolution patterns of attack chains are further disrupted, leading to reduced Recall for attack samples.

In summary, temporal encoding plays a critical role in improving Recall, enabling the model to identify attack behaviors spanning multiple time segments and reducing the likelihood of missed detections. In APT detection, Recall directly reflects the model’s coverage of attack samples. Therefore, the temporal encoding module is essential for ensuring detection completeness and capturing the dynamic nature of attack chains, preventing critical threats from remaining undetected over extended periods.

Experiment 2 (Without Node Masking)

Eliminating the node masking mechanism leads to a slight degradation in overall performance, with the most notable impact observed on Precision. On the Wget dataset, AUC decreases from 0.99 to 0.97 and F1-score from 0.97 to 0.95, while Recall remains relatively high at around 0.9. A similar trend is observed on CICAPT-IIoT, where Recall is maintained but Precision drops to 0.84. This indicates that without node masking, the model improves its ability to recall attack events but at the cost of increased false alarms, thereby reducing overall reliability.

The main function of node masking is to filter out redundant or noisy nodes, allowing the model to focus on more informative representations of critical nodes. By incorporating structural constraints, the mechanism combines node attributes with their topological context, producing more discriminative embeddings. Without this mechanism, the model treats all nodes indiscriminately, which increases coverage of attack samples but introduces noise, resulting in lower Precision and, consequently, a slight reduction in F1-score.

This phenomenon may be attributed to the following two factors: (i) in some datasets, node features lack sufficient discriminability, so the masking mechanism does not significantly emphasize critical nodes; and (ii) in smaller or simpler graphs, excessive filtering may remove valuable contextual information, limiting overall performance.

Overall, node masking contributes primarily to reducing false positives and improving detection precision. However, its overall performance gain is less pronounced compared with temporal encoding or edge reconstruction, making it more of an auxiliary optimization mechanism rather than a core component.

Experiment 3 (Without Edge Reconstruction)

The results demonstrate that the edge reconstruction module plays a central role in the overall framework. Eliminating this mechanism leads to a clear performance drop. On the Wget dataset, AUC decreases from 0.99 to 0.95 and F1-score from 0.97 to 0.93, while Precision and Recall decline to 0.91 and 0.96, respectively. Similar trends are observed on the CICAPT-IIoT dataset. This indicates that without edge reconstruction, both attack coverage and prediction accuracy are weakened, resulting in a significant reduction in overall detection capability.

The training loss curves further reveal this issue. After removing edge reconstruction, the loss drops sharply within a short period, showing fast convergence. However, this rapid decline does not signify improved capability; rather, it reflects overfitting to local node features due to the lack of structural constraints. As a result, the model quickly minimizes training loss but fails to generalize, with testing metrics deteriorating across the board—a phenomenon we term “false convergence”.

APT attacks inherently manifest in multi-stage interactions, where edge structures capture contextual dependencies and the evolving logic of attack chains. By guiding the model to reconstruct masked connections during training, edge reconstruction strengthens the model’s understanding of graph structures and enhances its ability to capture potential attack patterns. Without this mechanism, the model struggles to form robust representations at the global relational level, leaving temporal dependencies and node features unable to function effectively in the correct structural context, which leads to significant performance degradation.

In summary, the edge reconstruction module is not only crucial for maintaining overall performance but also essential for preventing overfitting and ensuring optimization stability during training. It is therefore an indispensable component of the proposed framework.

Experiment 4 (MAGIC)

From the experimental results, MAGIC demonstrates overall performance superior to most ablation variants on both datasets but remains slightly inferior to the complete framework. This indicates that even without explicit temporal modeling, the static dependency structures captured by its autoencoder architecture can still effectively represent attack patterns. Nevertheless, its precision and recall are consistently lower than those of the full model, suggesting that explicit temporal modeling further strengthens event-level sequential correlations and enhances the model’s ability to capture stage-dependent APT behaviors. For datasets such as Wget, where event sequences are continuous and behavioral patterns are well-defined, static structural information alone is sufficient to support high detection accuracy, though temporal modeling still plays a crucial role in ensuring global consistency and stability.

On the CICAPT-IIoT dataset, MAGIC achieves comparable average performance to the complete model but exhibits weaker stability and higher variance. This instability likely stems from the dataset’s inherent sparsity and limited event diversity. Without explicit temporal encoding, the model struggles to exploit time-dependent relationships in small-scale, weakly independent event sequences, thereby constraining its ability to distinguish contextual differences within the same attack stage and limiting its overall feature representation capacity.

Experiment 5 (Full Model)

The results serve as the baseline for comparison. On both datasets, the full model achieves the best or near-best performance across all metrics. On the Wget dataset, AUC reaches 0.99 and F1-score 0.97, while Precision and Recall remain high at 0.95 and 0.98, respectively. This demonstrates that the model can not only accurately identify attack samples but also maintain a low false alarm rate. On the CICAPT-IIoT dataset, although the overall values are slightly lower than those on Wget, the full model still delivers relatively optimal performance, particularly achieving a Recall of 0.97. This indicates that the model retains strong coverage capability even under limited data conditions.

Overall, the experimental results show that the full model, with temporal encoding, node masking, and edge reconstruction enabled, achieves a well-balanced trade-off between coverage and precision. Each module contributes in a complementary manner as follows: temporal encoding strengthens the sequential modeling of attack chains; node masking enhances the representation of key nodes, improving detection precision; and edge reconstruction significantly boosts the model's robustness in capturing complex relationships. Together, these components enable the full model to maintain stable and superior performance in complex APT detection scenarios within power systems.

5.5.2. Statistics and Visualization

To provide a more intuitive illustration of the impact of each module on model performance, we summarize the experimental results through statistical aggregation and present them with visualization charts. Compared with raw numerical comparisons, graphical representations more clearly reflect the variations of different metrics under various ablation settings, directly revealing the role and contribution of temporal encoding, node masking, and edge reconstruction in overall detection performance.

Figure 8 compares overall performance in terms of AUC and F1-score across the two datasets. Both exhibit the trend that the full model consistently outperforms the ablated variants, indicating that each module plays a positive role within the framework and that their combination provides complementary benefits, resulting in more robust detection outcomes.

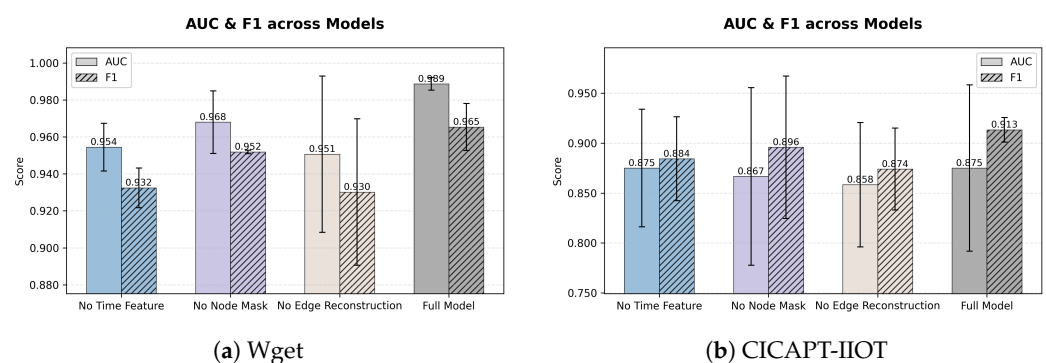


Figure 8. AUC and F1-Score of ablation models.

On the Wget dataset, the model achieves overall high performance, with the full model not only yielding the highest values but also the smallest fluctuations. This suggests that, with larger-scale data and more complex graph structures, the proposed approach can stably learn robust feature representations. In contrast, removing edge reconstruction leads to significantly larger fluctuations, reflecting the instability that arises when structural constraints are absent and the training process becomes more sensitive to noise.

On the CICAPT-IIoT dataset, the performance curves are noticeably lower, indicating that the model's discriminative power is constrained by its limited data scale and imbalanced sample distributions. Moreover, all ablated variants exhibit larger fluctuation ranges,

showing that results are more susceptible to randomness in small-sample scenarios and thus less stable. Nevertheless, the full model still maintains relative superiority, demonstrating that the joint design of the three modules ensures performance stability under diverse conditions.

Furthermore, Figure 9 illustrates the variation of F1-score with respect to Precision and Recall, providing an intuitive explanation of how F1 is determined. Both datasets reveal that F1 depends on the balance between Precision and Recall rather than either metric alone. The full model maintains the best trade-off between the two, consistently achieving the highest F1 performance. In comparison, removing temporal encoding results in a significant drop in Recall, leading to missed detections and lower F1; removing edge reconstruction degrades both Precision and Recall simultaneously, causing the most severe decline in F1, thereby confirming its critical role in overall performance. The effect of node masking is relatively minor, as the decrease in Precision is partially offset by the increase in Recall, resulting in only slight fluctuations in F1.

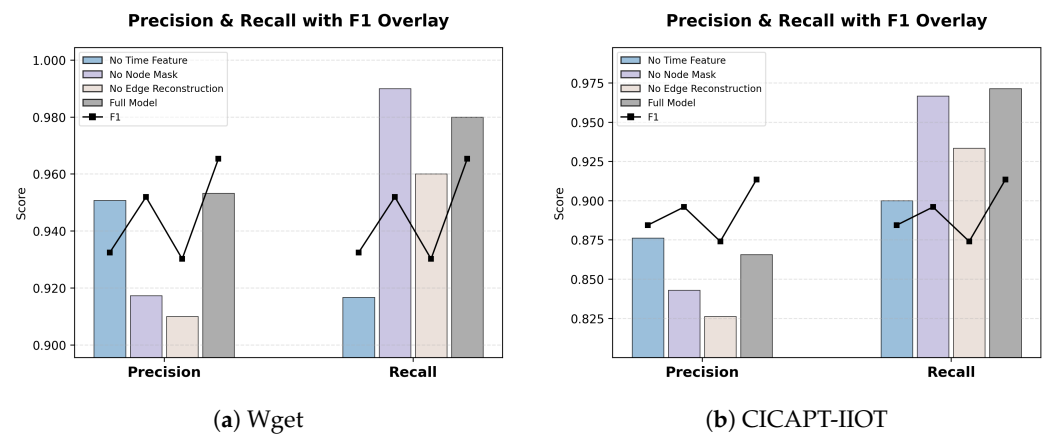


Figure 9. Precision, Recall, and F1 of ablation models.

As shown in the confusion matrix heatmaps (Figure 10), both datasets exhibit the pattern of a darker main diagonal and lighter off-diagonal entries. This indicates that the numbers of correctly classified samples (TP, TN) are relatively high, while false positives (FP) and false negatives (FN) remain comparatively low. Such a distribution directly reflects the model's favorable performance in terms of Precision and Recall as follows: a darker main diagonal suggests more comprehensive recognition of both benign and attack samples (high Recall), whereas lighter off-diagonal entries imply lower false-alarm and miss rates, thereby ensuring high detection accuracy (high Precision).

A closer comparison further reveals that the Wget dataset (Figure 10a) shows a sharper contrast between the main diagonal and off-diagonal regions, suggesting stronger recognition ability and fewer errors. Accordingly, both Precision and Recall remain at higher levels. In contrast, the CICAPT-IIoT dataset (Figure 10b) exhibits a weaker contrast, reflecting a lower proportion of correct classifications and increased false alarms and misses. This observation is consistent with its lower numerical scores and larger performance fluctuations.

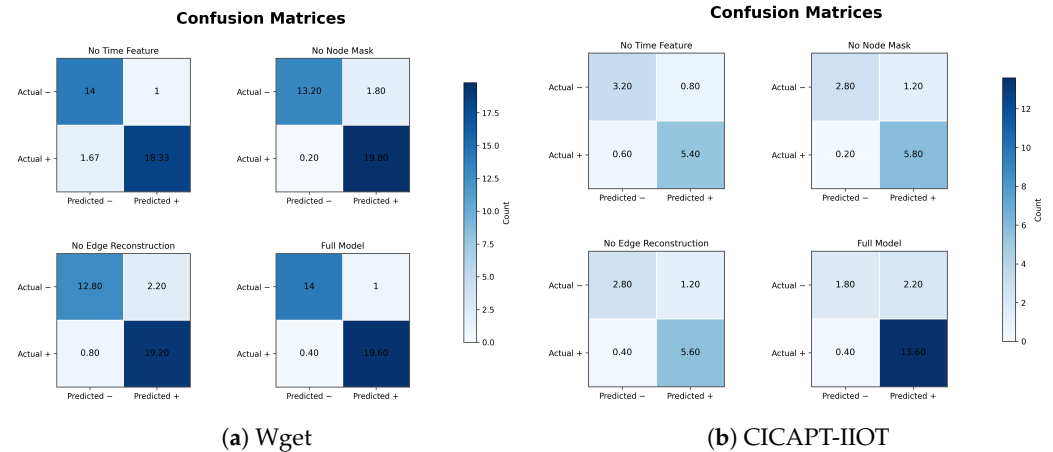


Figure 10. Confusion matrices.

To gain a more intuitive understanding of the latent features learned by the model, we present the dimensionality-reduced visualization of high-dimensional embeddings in Figure 11. Specifically, the latent space is projected onto a two-dimensional plane using both t-SNE and UMAP algorithms. Overall, the enhanced features exhibit clear inter-class separability in the low-dimensional space, where attack samples (red) and benign samples (gray) generally form relatively distinct cluster distributions.

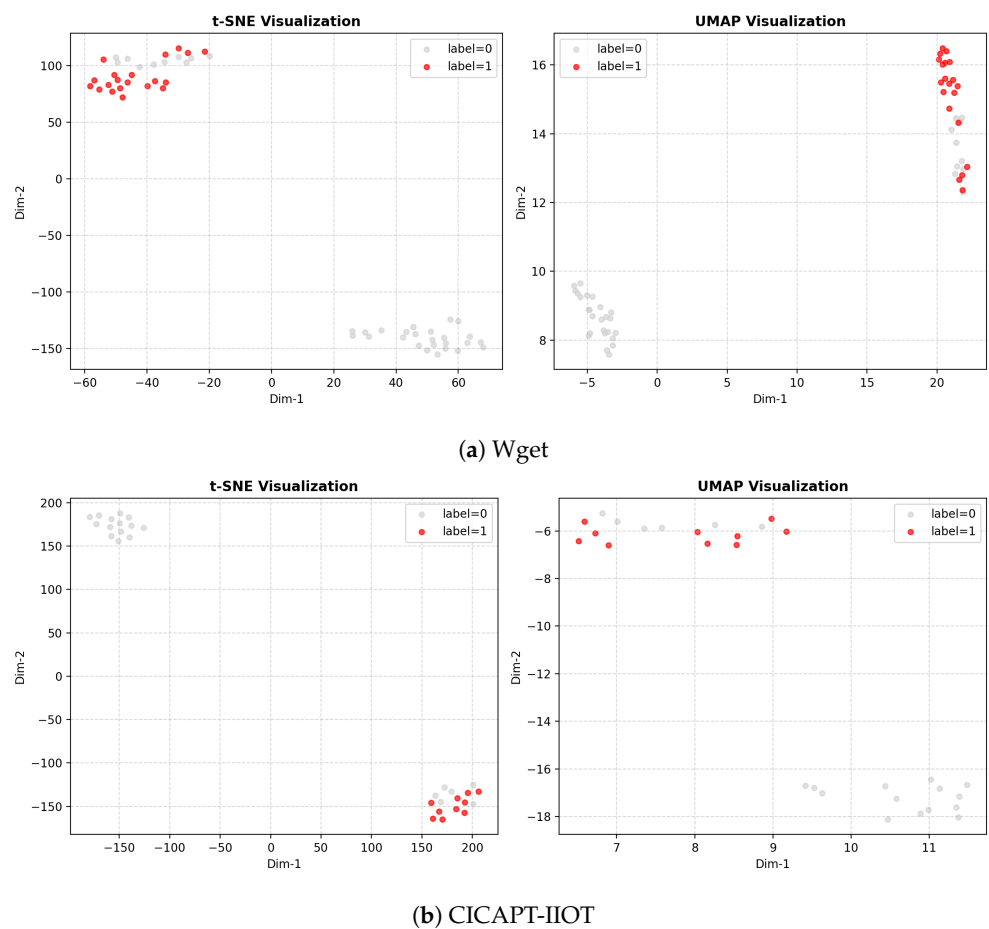


Figure 11. t-SNE and UMAP visualization.

More specifically, the Wget dataset demonstrates stronger inter-class separation with well-defined clustering structures, indicating that under larger-scale data and more complex graph structures, the enhanced features are more effectively learned and yield more robust

latent representations. In contrast, the distribution on the CICAPT-IIoT dataset appears looser and less distinct, showing relatively weaker separation.

It is noteworthy that benign samples tend to cluster densely within the same region, while malicious samples rarely appear within the benign cluster. This is because the model focuses on learning the intrinsic regularities of benign behavior during training, forming a high-density representation of the “normal pattern”. Samples that deviate significantly from this pattern—regardless of whether they correspond to known attack types—are mapped to regions distant from the benign cluster.

A small number of benign samples may, however, appear in anomalous regions. This typically occurs when certain normal operations, such as transient network fluctuations or maintenance activities, do not fully conform to the overall behavioral pattern. Since the model adopts a whitelist-based decision logic, any sample whose features fail to align with the benign distribution is classified as anomalous. This observation aligns with the model’s design philosophy, outlined as follows: the goal is to detect deviation from normality rather than to identify specific attack types. Consequently, even previously unseen or mutated attacks can be effectively detected, provided their behavioral characteristics differ substantially from the learned normal patterns.

6. Discussion

6.1. Analysis of Lower Performance on the CICAPT-IIOT Dataset

Experimental results show that the proposed model achieves higher detection accuracy on the Wget dataset, while its performance on the CICAPT-IIoT dataset is relatively lower, with most metrics decreasing by approximately 5–10%. This discrepancy primarily stems from the fundamental differences between the two datasets in terms of organizational structure and sample construction methodology.

First, the Wget dataset is inherently organized in the form of provenance graphs, where each sample contains a rich set of events and dependency relationships. After graph transformation, the number of nodes and edges per sample is roughly an order of magnitude larger than that of CICAPT-IIoT, enabling the model to more effectively capture structural dependencies and temporal correlations during training, thereby enhancing its representational capacity. In contrast, the CICAPT-IIoT dataset is not naturally structured as provenance graphs but consists of basic log events. To ensure there are sufficient samples for training and testing, event down-sampling was applied during graph construction, resulting in significantly fewer nodes per graph and limited structural information—consequently constraining the model’s ability to learn expressive features.

Second, each provenance graph in the Wget dataset corresponds to a complete and independent data collection process, fully representing a single APT attack episode. This independence allows the model to learn distinct and coherent attack patterns. By contrast, samples in CICAPT-IIoT are artificially constructed via temporal snapshot partitioning. While this method ensures an adequate number of samples, it introduces the following two limitations: (1) each sample often captures only a partial stage of the attack process, failing to represent the full behavioral sequence of an APT attack; (2) since all snapshots originate from the same continuous attack timeline, they exhibit intrinsic correlations in both attack stages and behavioral chains, thus lacking the inter-sample independence observed in Wget. These factors collectively account for the model’s performance degradation on the CICAPT-IIoT dataset.

Nevertheless, the CICAPT-IIoT dataset remains of great practical significance. Compared with Wget, it more closely mirrors the operational context of industrial control systems, thereby providing a more realistic representation of APT behaviors within power system environments. Hence, despite slightly lower quantitative results, the experiments

on CICAPT-IIoT substantiate the practical applicability and robustness of the proposed method in real-world power industry scenarios.

6.2. Future Work

In response to the performance disparities observed in the experiments, several potential directions for optimization and further research are proposed as follows:

- **Improvement of Snapshot Partitioning Strategy.** A more semantically consistent snapshot partitioning approach can be explored by incorporating event correlations, causal dependencies, and attack-stage information into the segmentation criteria. This would enhance the representational completeness of each sample while maintaining inter-sample independence, thereby avoiding information fragmentation caused by purely time-based partitioning.
- **Integration of Industry Data and Privacy-Preserving Collaboration Mechanisms.** Collaborations with power utilities or related industrial organizations could enable access to more domain-specific operational data, improving both the diversity and representativeness of training samples. Given the sensitivity of industrial data, privacy-preserving mechanisms such as federated learning, differential privacy, and secure multi-party computation may be employed to facilitate data sharing and collaborative model training without compromising confidentiality.
- **Graph-Based Data Augmentation and Synthetic Sample Generation.** To address the sparsity and structural simplicity of CICAPT-IIoT samples, graph augmentation and model-based data synthesis strategies can be adopted. Structural perturbation techniques such as node disturbance, subgraph fusion, and edge rewiring can enhance diversity at the topology level. Moreover, graph generation models, including GraphGAN and graph diffusion models, can be utilized to learn the distributional characteristics of original provenance graphs and automatically generate synthetic samples with similar topological and semantic properties. These approaches can effectively expand the training set without manual labeling, thereby improving the model's robustness and generalization capability under data-scarce scenarios.
- **Multidimensional Feature Mining and Information Enhancement.** Beyond increasing sample quantity, the efficiency of information utilization can be improved at the feature level. Although the CICAPT-IIoT dataset has relatively simple structural samples, its log records contain rich contextual attributes such as process ID, target object, file/network path, system call type, and event arguments. By performing finer-grained modeling and embedding of these features, the semantic density and discriminative power of the data can be significantly enhanced without increasing the number of samples.

7. Conclusions

In the domain of power systems as critical infrastructure, APT attacks are often stealthy and persistent, with attack chains spanning multiple time periods and system components, posing significant challenges to defense. This paper proposes a provenance graph-based feature enhancement method that integrates temporal encoding, node masking, and edge reconstruction. These mechanisms jointly improve temporal modeling capability, feature robustness, and structural representation at the graph level, enabling the detection model to effectively identify potential threats in complex and dynamic power network environments. Experimental results demonstrate that the proposed method significantly outperforms baseline approaches across multiple evaluation metrics, while the enhanced features exhibit strong inter-class separability and semantic clustering in low-dimensional visualizations.

Importantly, given the extreme scarcity of APT datasets in the power domain, we introduce a power-specific semantic mapping approach that deeply integrates generic provenance datasets with the operational context of power systems. This mapping not only improves the applicability and interpretability of the model in power scenarios but also provides a feasible pathway for developing domain-specific APT detection methods.

In summary, this study contributes both technically and practically to power system APT detection; it offers a novel methodological perspective for provenance graph modeling and provides valuable exploration in dataset construction and domain adaptation. These efforts lay the groundwork for advancing provenance graph-based intelligent detection in critical infrastructure security, and they offer theoretical and practical support for building more secure, interpretable, and deployable industrial network defense systems.

Author Contributions: Conceptualization, X.Z.; Methodology, X.Z.; Software, H.S.; Validation, H.S.; Formal analysis, X.Z.; Investigation, H.S.; Writing—original draft, X.Z.; Writing—review & editing, L.L.; Visualization, L.L.; Supervision, L.L.; Project administration, L.Z.; Funding acquisition, L.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The datasets used in this study are publicly available: Wget dataset at <https://github.com/margoseltzer/wget-apt/tree/master> (accessed on 19 October 2025) and CICAPT-IIoT dataset at <https://www.unb.ca/cic/datasets/iioT-dataset-2024.html> (accessed on 19 October 2025).

Conflicts of Interest: Authors Xuan Zhang, Haohui Su, Lincheng Li and Lvjun Zheng were employed by the company Extra High Voltage Power Transmission Company, China Southern Power Grid Co., Ltd., China. All authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Appendix A. Metrics

Appendix A.1. Confusion Matrix Components

The confusion matrix is the fundamental tool for binary classification performance analysis. It compares model predictions against ground-truth labels and categorizes samples into four types as follows:

TP (True Positive): Positive samples correctly predicted as positive.

TN (True Negative): Negative samples correctly predicted as negative.

FP (False Positive): Negative samples incorrectly predicted as positive.

FN (False Negative): Positive samples incorrectly predicted as negative.

Analyzing the distribution of TP, TN, FP, and FN provides insights into the model's false alarms and missed detections.

Appendix A.2. Precision

Precision measures the proportion of true positives among all samples predicted as positive, reflecting the accuracy of the model in identifying positive samples, as follows:

$$\text{Precision} = \frac{TP}{TP + FP}$$

A high Precision indicates a low false positive rate. In APT detection, high Precision means the alerts raised by the model are highly reliable.

Appendix A.3. Recall

Recall measures the proportion of true positives identified among all actual positive samples, reflecting the model's coverage of positive cases as follows:

$$\text{Recall} = \frac{TP}{TP + FN}$$

A Recall close to 1 indicates that the model can detect most true positives. In APT detection, high Recall ensures that more potential attack events are discovered, reducing missed detections.

Appendix A.4. F1-Score

The F1-score is the harmonic mean of Precision and Recall, providing a balanced measure of both as follows:

$$F_1 = 2 \cdot \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2TP}{2TP + FP + FN}$$

The F1-score ranges from 0 to 1. A higher value indicates that the model achieves both accurate and comprehensive detection, making it particularly suitable for evaluating performance under imbalanced data distributions.

Appendix A.5. AUC

The ROC curve plots the True Positive Rate (TPR) against the False Positive Rate (FPR) at different decision thresholds. AUC is the area under this curve, reflecting the classifier's overall discriminative capability across thresholds. AUC values closer to 1 indicate superior performance. From a probabilistic perspective, AUC can be interpreted as the probability that a randomly chosen positive sample is assigned a higher score than a randomly chosen negative sample, expressed as follows:

$$\text{AUC} = P(f(x^+) > f(x^-))$$

References

1. Akbarzadeh, A.; Erdodi, L.; Houmb, S.H.; Soltvedt, T.G. Two-stage advanced persistent threat (APT) attack on an IEC 61850 power grid substation. *Int. J. Inf. Secur.* **2024**, *23*, 2739–2758. [CrossRef]
2. IEC 61850; Communication Networks and Systems for Power Utility Automation. IEC: Geneva, Switzerland, 2013.
3. IEEE 1588-2019; Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems. IEEE: New York, NY, USA, 2019.
4. IEC 60870-5-104; Telecontrol Equipment and Systems—Part 5-104: Transmission Protocols—Network Access for IEC 60870-5-101 Using Standard Transport Profiles. IEC: Geneva, Switzerland, 2006.
5. Moreau, L.; Missier, P. PROV-DM: The PROV Data Model. W3C Recommendation REC-prov-dm-20130430, World Wide Web Consortium (W3C). 2013. Available online: <https://www.w3.org/TR/prov-dm/> (accessed on 23 August 2025).
6. Missier, P.; Dey, S.; Belhajjame, K.; Cuevas-Vicentín, V.; Ludäscher, B. D-PROV: Extending the PROV Provenance Model with Workflow Structure. In Proceedings of the 5th Workshop on the Theory and Practice of Provenance (TaPP 2013), Lombard, IL, USA, 2–3 April 2013.
7. Missier, P.; Bryans, J.W.; Gamble, C.; Curcin, V.; Danger, R. ProvAbs: Model, policy, and tooling for abstracting PROV graphs. In *Provenance and Annotation of Data and Processes, Proceedings of the 5th International Provenance and Annotation Workshop, IPAW 2014, Cologne, Germany, 9–13 June 2015*; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2015; Volume 8628, pp. 3–15. [CrossRef]
8. Firth, H.; Missier, P. ProvGen: Generating Synthetic PROV Graphs with Predictable Structure. In *Provenance and Annotation of Data and Processes, Proceedings of the 5th International Provenance and Annotation Workshop, IPAW 2014, Revised Selected Papers, Cologne, Germany, 9–13 June 2015*; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2015; Volume 8628, pp. 16–27. [CrossRef]
9. Wittner, R.; Mascia, C.; Gallo, M.; Frexia, F.; Müller, H.; Plass, M.; Geiger, J.; Holub, P. Lightweight Distributed Provenance Model for Complex Real-world Environments. *Sci. Data* **2022**, *9*, 503. [CrossRef] [PubMed]
10. Sembay, M.J.; de Macedo, D.D.J. Health Information Systems: Proposal of a Provenance Data Management Method in the Instantiation of the W3C PROV-DM Model. *Adv. Notes Inf. Sci.* **2022**, *2*, 192–201. [CrossRef]

11. Yusof, A.; Li, S.; Kawatra, A.S.; Li, D.; Chang, E.C.; Liang, Z. From Observations to Insights: Constructing Effective Cyberattack Provenance with PROVCON. In Proceedings of the Workshop on SOC Operations and Construction (WOSOC) 2025, San Diego, CA, USA, 24 February 2025. [\[CrossRef\]](#)
12. Kapoor, M.; Melton, J.; Ridenhour, M.; Moyer, T.; Krishnan, S. Flurry: A Fast Framework for Provenance Graph Generation for Representation Learning. In Proceedings of the 31st ACM International Conference on Information and Knowledge Management (CIKM 2022), Atlanta, GA, USA, 17–21 October 2022; pp. 4887–4891. [\[CrossRef\]](#)
13. Li, J.; Zhang, R.; Liu, J. ProvGRP: A Context-Aware Provenance Graph Reduction and Partition Approach for Facilitating Attack Investigation. *Electronics* **2024**, *13*, 100. [\[CrossRef\]](#)
14. Men, S.; Wang, J.; Ye, M. A PT-based Approach to Construct Efficient Provenance Graph for Threat Alert Investigation. In Proceedings of the 2023 5th International Conference on Advanced Information Science and System (AISS 2023), Virtual, 1–3 December 2024; Volume 60, p. 00016. [\[CrossRef\]](#)
15. Altinisik, E.; Deniz, F.; Sencar, H.T. ProvG-Searcher: A Graph Representation Learning Approach for Efficient Provenance Graph Search. In Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security (CCS 2023), Copenhagen, Denmark, 26–30 November 2023; pp. 2247–2261. [\[CrossRef\]](#)
16. Perozzi, B.; Al-Rfou, R.; Skiena, S. DeepWalk: Online Learning of Social Representations. In Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2014), New York, NY, USA, 24–27 August 2014; pp. 701–710. [\[CrossRef\]](#)
17. Grover, A.; Leskovec, J. node2vec: Scalable Feature Learning for Networks. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 2016; pp. 855–864. [\[CrossRef\]](#)
18. Han, X.; Pasquier, T.; Bates, A.; Mickens, J.; Seltzer, M. UNICORN: Runtime Provenance-Based Detector for Advanced Persistent Threats. In Proceedings of the Network and Distributed Systems Security Symposium (NDSS) 2020, San Diego, CA, USA, 13–17 August 2020; pp. 1–18. [\[CrossRef\]](#)
19. Kipf, T.N.; Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. In Proceedings of the 5th International Conference on Learning Representations (ICLR), Toulon, France, 24–26 April 2017.
20. Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; Bengio, Y. Graph Attention Networks. In Proceedings of the 6th International Conference on Learning Representations (ICLR 2018), Vancouver, BC, Canada, 30 April–3 May 2018.
21. Hamilton, W.L.; Ying, R.; Leskovec, J. Inductive Representation Learning on Large Graphs. In Proceedings of the Advances in Neural Information Processing Systems (NeurIPS) 30, Long Beach, CA, USA, 4–9 December 2017; pp. 1024–1034.
22. Li, Y.; Tarlow, D.; Brockschmidt, M.; Zemel, R. Gated Graph Sequence Neural Networks. In Proceedings of the 4th International Conference on Learning Representations (ICLR 2016), San Juan, Puerto Rico, 2–4 May 2016.
23. Jia, Z.; Xiong, Y.; Nan, Y.; Zhang, Y.; Zhao, J.; Wen, M. MAGIC: Detecting Advanced Persistent Threats via Masked Graph Representation Learning. In Proceedings of the 33rd USENIX Security Symposium (USENIX Security 24), Philadelphia, PA, USA, 14–16 August 2024; pp. 5197–5214.
24. Qiao, W.; Feng, Y.; Li, T.; Ma, Z.; Shen, Y.; Ma, J.F.; Liu, Y. Slot: Provenance-Driven APT Detection through Graph Reinforcement Learning. *arXiv* **2024**, arXiv:2410.17910. [\[CrossRef\]](#)
25. Bahar, A.A.M.; Ferrahi, K.S.; Messai, M.L.; Seba, H.; Amrouche, K. CONTINUUM: Detecting APT Attacks through Spatial-Temporal Graph Neural Networks. *arXiv* **2025**, arXiv:2501.02981. [\[CrossRef\]](#)
26. Zetter, K. Inside the Cunning, Unprecedented Hack of Ukraine’s Power Grid. *Wired*. 2016. Available online: <https://www.wired.com/2016/03/inside-cunning-unprecedented-hack-ukraines-power-grid/> (accessed on 28 August 2025).
27. Ghiasvand, E.; Ray, S.; Iqbal, S.; Dadkhah, S.; Ghorbani, A.A. CICAPT-IIoT: A provenance-based APT attack dataset for IIoT environment. *arXiv* **2024**, arXiv:2407.11278.
28. Xu, D.; Ruan, C.; Korpeoglu, E.; Kumar, S.; Achan, K. Inductive Representation Learning on Temporal Graphs. In Proceedings of the 8th International Conference on Learning Representations (ICLR 2020), Addis Ababa, Ethiopia, 26 April–1 May 2020.

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