SECBFF: TOPOLOGY-AWARE SECURITY LLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large Language Models (LLMs) are increasingly used to model temporal security event sequences, but they typically treat events as isolated textual streams and cannot exploit the interaction topology among the entities involved. In realistic security environments, this topology is valuable but incomplete: event streams evolve rapidly, new event types appear continuously, and graph representations are often available only for a small subset of stable entities through a frozen or third-party graph model. We introduce SECBFF, a topology-aware LLM framework designed for this low-coverage regime. The core of SECBFF is Topology-Injected Masked Fine-Tuning (TIM-FT), which teaches the LLM to condition on topological embeddings without expanding its vocabulary. SecBFF replaces the textual entity tokens for covered entities with embeddings from a frozen pre-trained Graph Foundation Model (GFM), projected into the LLM's embedding space, while uncovered entities retain their full textual representations. Because the GFM coverage of event entities is inherently sparse, the model naturally learns to exploit structural context when present without degrading when it is absent. Across three real-world datasets including enterprise security events, blockchain transactions, and cybersecurity events, and five temporal security detection tasks, SECBFF improves over diverse text-only and text-graph baselines, in some cases by large margins.We further provide a mechanistic analysis of how topology injection reshapes the model's internal computation, showing that GFM embeddings induce dedicated entity attention heads, maintain high-norm entity representations throughout the network, and form a persistent entity-to-entity communication subnetwork supporting a two-phase grounding-then-discrimination pattern for anomaly detection.

1 INTRODUCTION

Security monitoring over digital infrastructure requires analyzing temporal event sequences such as logins, transactions, and API calls to detect anomalous or malicious behavior (Chandola et al., 2009; Du et al., 2017). These events unfold over interaction graphs whose structure shapes what constitutes normal activity: a lateral move that is routine for an administrator may signal compromise from a guest account (Kent, 2016; Bowman et al., 2020). LLMs have recently emerged as powerful temporal reasoners for event streams, achieving strong results on log-based anomaly detection (Guan et al., 2024; Qi et al., 2023) and security event classification (Ferrag et al., 2024). Yet these methods treat each sequence in isolation, discarding the topological context of who interacts with whom that often separates benign from malicious behavior.

GFMs can encode this topological context through pretrained entity embeddings (Bechler-Speicher et al., 2026; Liu et al., 2025), but integrating them with LLMs for temporal security detection is an open problem. Prior graph–LLM work projects graph representations into the token embedding space for tasks such as node classification and graph question answering (Perozzi et al., 2024; Liu et al., 2024; Wang et al., 2024), but these methods target static graph reasoning and do not address evolving event streams or the temporal dynamics of security data. More fundamentally, graph coverage in security environments is often incomplete: new entities and event types appear continuously (Apruzzese et al., 2023), so GFM embeddings may exist for only a subset of stable entities. A practical framework must exploit structural signals where available, degrade gracefully to text-only reasoning elsewhere, and avoid frequent vocabulary expansion or retraining.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

We introduce SECBFF (Security Billion-scale Foundation model Fusion), a framework that bridges GFMs and LLMs for temporal security detection under this partial-coverage regime. Its core mechanism is Topology-Injected Masked Fine-Tuning (TIM-FT): pretrained GFM embeddings are projected into the LLM's input-embedding space and *replace* the textual entity tokens for covered entities, while uncovered entities retain their full textual representations. Unlike prior graph–LLM alignment approaches that prepend graph summary tokens as a prefix (Perozzi et al., 2024; Liu et al., 2024; Wang et al., 2024) or require every entity to fall within the graph model's node and edge types, TIM-FT operates entirely in continuous representation space, preserving the sequence structure and letting the LLM combine open-vocabulary temporal reasoning with sparse topological signals without discrete entity tokenization or vocabulary changes. Because graph coverage is inherently sparse, the model naturally learns to exploit graph context when present without overfitting to its availability.

Across three real-world datasets, enterprise security logs, Elliptic Bitcoin transactions (Weber et al., 2019), and LANL cybersecurity events (Kent, 2015), and five detection tasks, SECBFF outperforms text-only and text-graph baselines, in some cases by large margins, including in zero-shot and long-sequence settings. It remains effective even when topological coverage is sparse (15–25% of entities), and its text-first design preserves flexibility for previously unseen event types. Beyond predictive performance, we conduct a mechanistic analysis of how topology injection reshapes the LLM's internal computation. We find that GFM embeddings induce dedicated entity attention heads, sustain high-norm entity representations across all layers (10–60$\times$ those of a text-only model), and form a persistent entity-to-entity communication subnetwork that supports a two-phase grounding-then-discrimination mechanism for anomaly detection, none of which emerge in an identically trained text-only model.

Our main contributions are:

1. We introduce SECBFF, a topology-aware framework for temporal security detection that integrates LLMs with GFMs under partial topological coverage.
2. We propose Topology-Injected Masked Fine-Tuning (TIM-FT), a strategy for conditioning LLMs on continuous topological embeddings without vocabulary expansion, gracefully handling entities with and without graph coverage.
3. We demonstrate state-of-the-art performance across five security tasks and three datasets, showing robustness to sparse topology, varying sequence lengths, and zero-shot transfer.
4. We provide a mechanistic analysis characterizing how topology injection reshapes the LLM's computation.

2 RELATED WORK

Graph-based security and GFMs. Graph neural networks have been widely applied to security tasks where entities and their interactions form a natural graph structure, including financial fraud detection on transaction graphs (Weber et al., 2019; Cheng et al., 2025) and lateral movement detection in enterprise networks (Bowman et al., 2020; Liu et al., 2018). These methods leverage relational structure effectively but operate on fixed graph snapshots and do not model the temporal evolution of event sequences. More recently, GFMs have scaled graph representation learning to large heterogeneous graphs, producing transferable entity embeddings that encode persistent structural and semantic properties (Bechler-Speicher et al., 2026; Liu et al., 2025; Mao et al., 2024). SECBFF treats such embeddings as frozen side information, avoiding the cost of joint graph-LLM training while benefiting from the structural priors learned during graph pretraining.

Graph–LLM alignment and continuous conditioning. A broad line of work conditions language models on continuous vectors rather than discrete vocabulary items, including prefix-tuning (Li & Liang, 2021), vision-language alignment via frozen encoders (Tsimpoukelli et al., 2021), retrieval compression into dense latent tokens (Cheng et al., 2024), and reasoning in continuous latent space (Hao et al., 2025). A parallel direction aligns graph representations with LLMs by mapping graph features into the token embedding space: GraphToken (Perozzi et al., 2024) extends prompts with graph-derived continuous tokens, GraphPrompter (Liu et al., 2024) uses GNN-derived soft prompts, TEA-GLM (Wang et al., 2024) projects GNN representations into LLM token em-

beddings, and RGLM (Zhang et al., 2026) adds reconstructive graph supervision. However, these methods target static graph reasoning tasks such as node classification and graph question answering, rather than temporal anomaly detection over evolving event streams.

LLMs for security event analysis. Recent work applies LLMs to log-based anomaly detection. LogLLM (Guan et al., 2024) uses BERT to encode log messages into semantic vectors aligned to a LLaMA decoder through a projector, and LogGPT (Qi et al., 2023) prompts ChatGPT directly for log-based anomaly detection without task-specific alignment. These methods demonstrate that LLMs can serve as effective temporal reasoners for security events but rely exclusively on textual representations and do not incorporate structural information from entity interaction graphs. SECBFF bridges these lines of work by injecting GFM embeddings into an LLM for temporal security detection under partial topological coverage, a regime not addressed by existing approaches.

3 METHOD

We propose SECBFF, a framework for temporal security reasoning that conditions an LLM on graph-derived entity representations. The event sequence remains the primary input; for entities within events that have pretrained graph embeddings, SECBFF replaces the entity's textual tokens with the projected graph embedding directly in the LLM's input-embedding space. Entities without graph coverage retain their full textual representation. During fine-tuning, available topology injection is applied only for the sparse subset of entities with graph coverage, so the model naturally learns to exploit structural context when present without degrading when it is absent.

Problem setup. A security event sequence is a temporally ordered tuple $s = (e_1, \ldots, e_T)$. Each event e_t carries a textual serialization x_t, a natural-language or template-based description of the action that includes mentions of one or more participating entities. We denote the set of entities mentioned in event e_t by $\mathcal{U}_t$; these may correspond to users, devices, accounts, services, contracts, or other domain-specific objects, each mapping to a node or edge in an underlying interaction graph $G = (V, E)$.

We assume access to a pretrained graph model that provides embeddings at both node and edge granularity:

$$f_G : V_G \cup E_G \;\rightarrow\; \mathbb{R}^{d_g},$$

where $V_G \subseteq V$ and $E_G \subseteq E$ are the subsets of nodes and edges for which an embedding is available. This formulation covers two practical regimes: (i) a frozen GFM that can be queried but not fine-tuned, and (ii) a fixed table of pretrained embeddings. In both cases, SECBFF treats graph representations as fixed side information and trains only the LLM-side parameters.

Topological coverage is partial by design. For a given event e_t, the *covered* entities are $\widetilde{\mathcal{U}}_t = \mathcal{U}_t \cap (V_G \cup E_G)$; the remaining entities in $\mathcal{U}_t \setminus \widetilde{\mathcal{U}}_t$ are *uncovered* and represented through text alone. Our goal is to learn a model that reasons over the event sequence s while conditioning on available topological context wherever it exists.

3.1 TOPOLOGY-INJECTED MASKED FINE-TUNING

Graph embedding retrieval and projection. For each covered entity $u \in \widetilde{\mathcal{U}}_t$, we retrieve its embedding $z_u = f_G(u) \in \mathbb{R}^{d_g}$ and map it into the LLM embedding space via a learned projector:

$$P_\theta : \mathbb{R}^{d_g} \rightarrow \mathbb{R}^{d_\ell}, \qquad p_u = P_\theta(z_u),$$

where d_ℓ is the LLM token-embedding dimension. We use a single linear projection followed by layer normalization for simplicity. The resulting vector p_u is a *topology token*: a continuous input embedding that carries structural information but has no discrete vocabulary entry.

Topology injection by replacement. Let $x_t = (w_{t,1}, \ldots, w_{t,L_t})$ be the tokenized text of event e_t, with LLM input embeddings $E(w_{t,j}) \in \mathbb{R}^{d_\ell}$. Each entity mention $u \in \mathcal{U}_t$ spans one or more tokens in x_t. For each covered entity $u \in \widetilde{\mathcal{U}}_t$, SECBFF replaces its mention span with the corresponding topology token p_u at the embedding level. The remaining tokens, including action descriptions,

timestamps, contextual fields, and uncovered entity mentions, are preserved unchanged. Since the embedding replacement operates before the first attention layer, the method is compatible with standard inference optimizations such as KV-caching and continuous batching. The augmented event representation is

$$\tilde{x}_t = \mathrm{Replace}(x_t,\ \{p_u : u \in \widetilde{\mathcal{U}}_t\}),$$

Replacement is performed *in place*, at every occurrence of the entity in the sequence: a mention spanning m tokens is replaced by m copies of the topology token p_u, so the number of input positions, the positional indices of all surrounding tokens, and the number of visible events are identical to those of the text-only baseline. SECBFF therefore does not compress the sequence, and its gains cannot be attributed to a larger effective context or to more events fitting within the context window (see Appendix M). The full LLM input is the temporal concatenation

$$X = [\tilde{x}_1; \ldots; \tilde{x}_T].$$

Because injection operates at the embedding level through replacement, the tokenizer and output vocabulary remain unchanged. The autoregressive loss is computed only over textual token positions; positions occupied by topology tokens are excluded from the loss, so the model is never asked to predict a graph embedding.

Training objective. Let $\widetilde{X}$ denote the full input sequence after topology injection, and let $\mathcal{I}_{\text{text}} \subseteq \{1, \ldots, N\}$ denote the positions corresponding to textual tokens, excluding topology-token positions. We fine-tune the LLM using the causal language modeling loss over textual positions only:

$$\mathcal{L} = -\sum_{i \in \mathcal{I}_{\text{text}}} \log\, p_\Theta(w_i \mid w_{<i},\, \widetilde{X}),$$

where w_i is the token at position i and Θ comprises the projector P_θ and the LLM adaptation parameters (LoRA weights in our implementation). The graph model f_G remains frozen throughout. Because topology tokens replace entity mentions at the embedding level, the causal structure of the sequence is preserved: each textual token is predicted from all preceding tokens and topology tokens, but topology-token positions are excluded from the loss. The model is never asked to predict a graph embedding or reconstruct GFM representations; topology tokens serve exclusively as conditioning variables for textual prediction.

Robustness to partial coverage arises naturally from this formulation. In practice, GFM coverage is sparse. The model therefore encounters uncovered entities in every training sequence and learns to condition on topology tokens when they are present while relying on the textual entity description otherwise. No additional stochastic masking is required: the inherent sparsity of graph coverage acts as an implicit regularizer, preventing the model from depending exclusively on the graph signal.

4 EXPERIMENTS

4.1 DATASETS

We evaluate SECBFF on three datasets spanning enterprise cybersecurity, cryptocurrency fraud, and national laboratory intrusion detection. Each dataset pairs a temporal sequence of behavioral events with an underlying entity interaction graph that may cover only a fraction of the entities appearing in the event logs; we pre-train a GFM on each graph using the GraphBFF framework (Bechler-Speicher et al., 2026) to produce 128-dimensional node embeddings that serve as the structural signal injected into the LLM.

Enterprise Security Events. Enterprise security event logs from a production system spanning February 2026, with 11,411 users and 974,724 events split 80/20 into training and test users. Each record is serialized as a short natural-language string naming the actor, the timestamp, the action, and the target entity, for example `User_X [timestamp] accessed Entity_Y`. The dataset contains labeled anomalies of three types: novel entity access, temporal burst, and behavioral type shift, comprising approximately 7.5% of events overall. The entity interaction graph comes from a production GFM pre-trained on a social and financial interaction network that is completely disjoint from the downstream evaluation graph, providing 128-dimensional embeddings for approximately 25% of entities.

Table 1: Evaluation on Enterprise Security Events, across full-sequence, real-time, zero-shot real-time, one-event zero-shot, and event-localization settings. We report AUC-PR for classification and MRR for localization; higher is better for all metrics. Results are mean ± standard deviation over three random seeds, rounded to two decimal places.

	Method	Sequence AUC-PR	Real-Time AUC-PR	Zero-Shot RT AUC-PR	One-Event ZS RT AUC-PR	Event MRR
Abl.	No FT	$0.711_{\pm 0.02}$	$0.610_{\pm 0.02}$	$0.555_{\pm 0.03}$	$0.184_{\pm 0.02}$	$0.575_{\pm 0.02}$
	Random-GFM	$0.371_{\pm 0.03}$	$0.676_{\pm 0.01}$	$0.545_{\pm 0.03}$	$0.236_{\pm 0.02}$	$0.350_{\pm 0.02}$
Text	Text-only	$0.949_{\pm 0.01}$	$0.833_{\pm 0.01}$	$0.833_{\pm 0.01}$	$0.631_{\pm 0.02}$	$0.925_{\pm 0.01}$
	LogLLM	$0.471_{\pm 0.04}$	$0.869_{\pm 0.01}$	$0.576_{\pm 0.02}$	$0.264_{\pm 0.02}$	$0.450_{\pm 0.03}$
	Text-graph	$0.271_{\pm 0.04}$	$0.856_{\pm 0.01}$	$0.572_{\pm 0.02}$	$0.254_{\pm 0.02}$	$0.920_{\pm 0.01}$
Graph+Text	GraphPoolMLP	$0.930_{\pm 0.01}$	$0.451_{\pm 0.03}$	$0.415_{\pm 0.03}$	$0.173_{\pm 0.02}$	$0.616_{\pm 0.02}$
	GraphPoolLinear	$0.968_{\pm 0.01}$	$0.885_{\pm 0.01}$	$0.475_{\pm 0.02}$	$0.158_{\pm 0.02}$	$0.926_{\pm 0.01}$
	GraphToken	$0.968_{\pm 0.01}$	$0.866_{\pm 0.01}$	$0.478_{\pm 0.02}$	$0.156_{\pm 0.01}$	$0.928_{\pm 0.01}$
	GraphPrompter	$0.943_{\pm 0.01}$	$0.731_{\pm 0.02}$	$0.446_{\pm 0.03}$	$0.161_{\pm 0.02}$	$0.921_{\pm 0.01}$
	SecBFF	$\mathbf{0.992}_{\pm \mathbf{0.00}}$	$\mathbf{0.893}_{\pm \mathbf{0.01}}$	$\mathbf{0.872}_{\pm \mathbf{0.01}}$	$\mathbf{0.733}_{\pm \mathbf{0.02}}$	$\mathbf{0.954}_{\pm \mathbf{0.01}}$

Elliptic Bitcoin. The Elliptic Bitcoin dataset (Weber et al., 2019) contains 203,769 Bitcoin transactions with 4,545 illicit and 42,019 licit labeled transactions across 49 timesteps. We use a temporal split: timesteps 1–34 for graph pre-training and 35–49 for LLM fine-tuning and evaluation. Event sequences are constructed via breadth-first ego-graph expansion, yielding 11,973 sequences with approximately 5.9% illicit. GFM embeddings are available for approximately 15% of entities.

LANL 2015. The LANL 2015 dataset (Kent, 2015) contains authentication, network flow, process, and DNS logs from Los Alamos National Laboratory with ground-truth red-team labels. We construct a temporal interaction graph with 449,351 nodes and 290,852 edges, using hourly buckets 0–4 for graph pre-training and buckets 5 onward for evaluation (2.8% anomaly rate). GFM embeddings are available for approximately 15% of entities. Full dataset construction details are provided in Appendix F, and per-dataset summary statistics in Appendix J.

4.2 Setting

All methods share the same self-supervised fine-tuning protocol: LoRA adaptation on LLaMA 3 8B with next-token prediction on event sequences. For the enterprise dataset, only normal sequences are used during this phase. For Elliptic and LANL, all available sequences are used regardless of label, as the self-supervised objective does not use labels and the illicit/compromised fraction is small. For SecBFF, this fine-tuning uses TIM-FT with topology injection for covered entities; for all other methods, each injects topology through its own mechanism during the same fine-tuning phase. Only training-split sequences are used in this phase; evaluation sequences are never seen during self-supervised fine-tuning. After self-supervised fine-tuning, a task-specific classification adapter is trained on labeled data. All reported numbers are the mean ± standard deviation over three seeds, with every baseline rerun under identical seeds, search, and compute budgets. Full hyperparameters and classification adapter details are provided in Appendix A, and further protocol notes in Appendix M.

4.3 Evaluation Protocol

We evaluate across five settings:

- **Sequence classification** determines whether a complete user event sequence contains any anomaly.
- **Real-time detection** simulates a growing event stream by creating prefix samples ending at different events and classifying whether the final event is anomalous.

Table 2: Evaluation on Elliptic Bitcoin dataset, across full-sequence, real-time, zero-shot real-time, one-event zero-shot, and event-localization settings. We report AUC-ROC for classification and MRR for localization; higher is better for all metrics. AUC-PR for the same settings is reported in Appendix D. Results are mean ± standard deviation over three random seeds, rounded to two decimal places.

	Method	Sequence AUC-ROC	Real-Time AUC-ROC	Zero-Shot RT AUC-ROC	One-Event ZS RT AUC-ROC	Event MRR
Abl.	No FT	$0.484_{\pm 0.03}$	$0.545_{\pm 0.02}$	$0.474_{\pm 0.03}$	$0.478_{\pm 0.03}$	$0.312_{\pm 0.02}$
	Random-GFM	$0.598_{\pm 0.02}$	$0.599_{\pm 0.02}$	$0.509_{\pm 0.03}$	$0.523_{\pm 0.02}$	$0.385_{\pm 0.03}$
Text	Text-only	$0.741_{\pm 0.01}$	$0.735_{\pm 0.02}$	$0.529_{\pm 0.03}$	$0.533_{\pm 0.02}$	$0.518_{\pm 0.02}$
	LogLLM	$0.738_{\pm 0.02}$	$0.766_{\pm 0.01}$	$0.521_{\pm 0.03}$	$0.545_{\pm 0.02}$	$0.503_{\pm 0.03}$
	Text-graph	$0.748_{\pm 0.01}$	$0.735_{\pm 0.02}$	$0.479_{\pm 0.04}$	$0.571_{\pm 0.02}$	$0.549_{\pm 0.02}$
Graph+Text	GraphPoolMLP	$0.579_{\pm 0.03}$	$0.494_{\pm 0.03}$	$0.516_{\pm 0.02}$	$0.511_{\pm 0.03}$	$0.401_{\pm 0.03}$
	GraphPoolLinear	$0.559_{\pm 0.03}$	$0.609_{\pm 0.02}$	$0.521_{\pm 0.03}$	$0.477_{\pm 0.03}$	$0.376_{\pm 0.03}$
	GraphToken	$0.722_{\pm 0.01}$	$0.781_{\pm 0.02}$	$0.542_{\pm 0.02}$	$0.502_{\pm 0.02}$	$0.573_{\pm 0.02}$
	GraphPrompter	$0.741_{\pm 0.02}$	$0.745_{\pm 0.02}$	$0.533_{\pm 0.03}$	$0.489_{\pm 0.03}$	$0.564_{\pm 0.02}$
	SecBFF	$\mathbf{0.796}_{\pm \mathbf{0.01}}$	$\mathbf{0.792}_{\pm \mathbf{0.01}}$	$\mathbf{0.588}_{\pm \mathbf{0.02}}$	$\mathbf{0.598}_{\pm \mathbf{0.02}}$	$\mathbf{0.627}_{\pm \mathbf{0.02}}$

- **Zero-shot real-time detection** applies the same prefix evaluation without any task-specific fine-tuning, using only the self-supervised backbone.
- **One-event zero-shot detection** gives the model a single event, which it must classify without any preceding context.
- **Event localization** ranks all events in a known-anomalous sequence by per-event negative log-likelihood to pinpoint which events are anomalous. This setting is evaluated on all three datasets.

We report AUC-PR for the enterprise dataset, where the anomaly rate is synthetically controlled (each candidate suffix event is injected with probability 0.15, giving $\approx 7.5\%$ anomalous events overall; see Appendix C), and AUC-ROC for Elliptic and LANL, where the class distribution reflects the original datasets. AUC-PR values for Elliptic and LANL are additionally reported in Appendix D. For localization we report mean reciprocal rank (MRR). The prompt templates used for each setting are listed in Appendix G.

4.4 Baselines

We compare SecBFF against nine baselines organized into three categories (Table 4): text-only methods that use no graph information (No FT, Text-only, Text-graph), embedding-based methods that inject non-graph-native embeddings (Random-GFM, LogLLM (Guan et al., 2024)), and graph-aware methods that inject GFM-derived representations as prefix tokens (GraphPoolLinear, GraphPoolMLP, GraphToken (Perozzi et al., 2024), GraphPrompter (Liu et al., 2024)). All variants share the same LLaMA 3 8B backbone and LoRA configuration ($\sim$13.6M parameters); projector parameters are additional. Full baseline descriptions are provided in Appendix B. Random-GFM is a controlled ablation rather than a competing method: because the projector's layer normalization fixes the injected-token norm regardless of its input, it holds norm, injection position, and coverage constant and varies only the *content* of the injected vectors, isolating the learned graph geometry as the source of any gain (Section 4.7).

Detectors that rely on the graph as the predictor (temporal and static GNNs, graph anomaly detection, and classical log-based methods) are handicapped by construction in this regime, since they have no representation for the majority of events and cannot score an entity that no graph build yet covers. We evaluate seven such detectors in Appendix E for completeness; all perform at or near chance on the full stream.

4.5 RESULTS

SECBFF achieves the best performance across all five evaluation settings on all three datasets (15 out of 15 configurations). The gains are largest in low-context settings such as zero-shot and one-event detection, where topological priors compensate for limited textual context. We now discuss each dataset in detail.

Enterprise security events. SECBFF achieves the highest score in every setting (Table 1). The Random-GFM ablation is the most informative comparison: replacing real embeddings with random vectors does not merely remove the benefit but drives sequence classification below the un-fine-tuned model, so injecting uninformative vectors at entity positions is actively harmful rather than simply unhelpful.

The advantage of topology is most pronounced where textual context is thin. SECBFF improves on the text-only baseline by 4.7% on zero-shot real-time detection, 16.2% on one-event detection, and 3.1% on event-localization MRR. The prefix-based alternatives degrade sharply under zero-shot transfer despite competitive in-distribution performance: GraphPoolLinear is within three points of SECBFF on sequence classification yet trails it by 45% relative on zero-shot real-time, indicating that in-place replacement yields more transferable representations than prepending graph summary tokens. This is not a context-budget effect: replacement is in place and length-preserving, so every graph-aware baseline consumes strictly more context than SECBFF does. Text-graph and LogLLM fall below even the text-only baseline on sequence classification, since serializing graph neighborhoods or compressing them through a text encoder disrupts the temporal signal that sequence-level judgments depend on, though both recover on the real-time setting where the prefix evaluation provides a more focused context window.

Elliptic Bitcoin. Elliptic is notably harder, with all methods closer to chance under zero-shot transfer, yet SECBFF is again best in all five settings (Table 2). The prefix-based methods again lose most of their advantage under zero-shot transfer, consistent with the enterprise finding that prefix fusion overfits to the in-distribution setting. Random-GFM again costs 25% of sequence AUC-ROC, confirming that what matters is the content of the injected embedding rather than its presence.

LANL 2015. On LANL 2015, which carries ground-truth red-team labels, SECBFF is best in all five settings (Table 3), outperforming the strongest baseline, GraphPrompter, by 13.3% on sequence classification and the text-only baseline by 18.8%. Zero-shot detection remains hard for every method, but SECBFF still leads one-event zero-shot by 8.6% over the next best. Random-GFM again degrades sharply, and the prefix-based variants again fall below text-only on the sequence and real-time settings, consistent with the pattern across all three datasets.

4.6 ROBUSTNESS TO TOPOLOGICAL COVERAGE

The results above use each dataset's *natural* coverage rate (15–25%). Because coverage is the central assumption of our setting, we control it directly on Elliptic, sweeping exposed coverage from 3% to 25% and separately testing mismatch between training and inference coverage. SECBFF degrades gracefully as topology becomes scarce, landing within one standard deviation of the text-only baseline at 3% coverage, so topology too sparse to help does not measurably hurt. Coverage shift is modest but asymmetric, giving the recommendation to train at or below the coverage expected at inference. The full sweep and shift study are given in Appendix H.

4.7 HOW DO GFM EMBEDDINGS TRANSFORM LLM BEHAVIOR?

The results above show *that* SECBFF outperforms baselines, but not *how* it does so. To understand the mechanism, we compare the internal representations of SECBFF against the identically trained text-only baseline on 200 enterprise sequences, probing attention patterns and hidden-state dynamics across all 32 layers of the LLM.

Entity norm dynamics. Figure 1 shows the mean L_2 norm of entity hidden states at each layer. In SECBFF, entity norms begin at $\sim$64 at layer 0. This starting value is *not* evidence of anything: the projector ends in a layer normalization, which produces an expected norm of $\sqrt{d_\ell} = 64$ for

Table 3: Evaluation on LANL 2015 Cybersecurity dataset, across full-sequence, real-time, zero-shot real-time, one-event zero-shot, and event-localization settings. We report AUC-ROC for classification and MRR for localization; higher is better for all metrics. AUC-PR for the same settings is reported in Appendix D. Results are mean ± standard deviation over three random seeds, rounded to two decimal places.

	Method	Sequence AUC-ROC	Real-Time AUC-ROC	Zero-Shot RT AUC-ROC	One-Event ZS RT AUC-ROC	Event MRR
Abl.	No FT	$0.512_{\pm 0.03}$	$0.601_{\pm 0.03}$	$0.482_{\pm 0.04}$	$0.493_{\pm 0.03}$	$0.328_{\pm 0.03}$
	Random-GFM	$0.673_{\pm 0.02}$	$0.774_{\pm 0.02}$	$0.516_{\pm 0.03}$	$0.514_{\pm 0.02}$	$0.419_{\pm 0.03}$
Text	Text-only	$0.723_{\pm 0.02}$	$0.786_{\pm 0.01}$	$0.518_{\pm 0.03}$	$0.495_{\pm 0.03}$	$0.467_{\pm 0.02}$
	LogLLM	$0.685_{\pm 0.03}$	$0.742_{\pm 0.02}$	$0.547_{\pm 0.03}$	$0.521_{\pm 0.03}$	$0.438_{\pm 0.04}$
	Text-graph	$0.721_{\pm 0.02}$	$0.782_{\pm 0.01}$	$0.487_{\pm 0.04}$	$0.506_{\pm 0.03}$	$0.481_{\pm 0.02}$
Graph+Text	GraphPoolMLP	$0.566_{\pm 0.03}$	$0.525_{\pm 0.04}$	$0.503_{\pm 0.03}$	$0.546_{\pm 0.02}$	$0.372_{\pm 0.03}$
	GraphPoolLinear	$0.526_{\pm 0.04}$	$0.559_{\pm 0.03}$	$0.541_{\pm 0.03}$	$0.534_{\pm 0.02}$	$0.365_{\pm 0.04}$
	GraphToken	$0.730_{\pm 0.02}$	$0.785_{\pm 0.02}$	$0.539_{\pm 0.03}$	$0.528_{\pm 0.03}$	$0.492_{\pm 0.02}$
	GraphPrompter	$0.758_{\pm 0.02}$	$0.791_{\pm 0.02}$	$0.527_{\pm 0.03}$	$0.514_{\pm 0.03}$	$0.503_{\pm 0.02}$
	SecBFF	$\mathbf{0.859}_{\pm \mathbf{0.01}}$	$\mathbf{0.828}_{\pm \mathbf{0.01}}$	$\mathbf{0.574}_{\pm \mathbf{0.02}}$	$\mathbf{0.593}_{\pm \mathbf{0.02}}$	$\mathbf{0.618}_{\pm \mathbf{0.02}}$

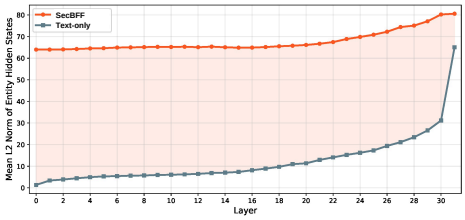

Figure 1: Entity representation norms across layers. SECBFF maintains 10–60× higher norms than the text-only model, a gap actively preserved by the network. The layer-0 value is fixed by the projector's layer normalization; the trajectory is not.

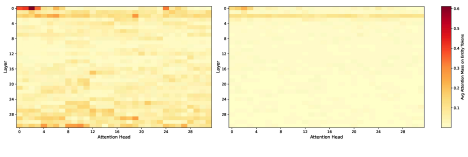

Figure 2: Attention head entity specialization. SECBFF (left) develops dedicated entity heads with up to 0.6 attention mass; text-only (right) shows none.

$d_\ell = 4096$ regardless of what is fed into it. Random-GFM, which shares the projector, starts from the same norm. The informative observation is what happens afterwards. In SECBFF the norm is maintained and amplified across all 32 layers, reaching ∼80 by layer 31 rather than decaying toward the text-token norm, whereas in the text-only model entity norms start near zero and reach only ∼17 by layer 26. Because the layer-0 norm is fixed by the architecture, the divergence in *trajectory* is attributable to the learned LoRA parameters and the content they operate on, not to the projector. The network is choosing to preserve this 10–60× gap, and it is that choice, rather than the initial scale, that is functionally meaningful.

Dedicated entity heads. Figure 2 shows that SECBFF develops dedicated "entity heads" with attention mass up to 0.6 directed at entity positions, concentrated in layers 0–4 and 24–31, while the text-only model shows uniformly low entity attention (∼0.05–0.1) with no specialization. High-norm representations produce larger query–key dot products, so sustained norms plausibly supply the *capacity* for such specialization to form. We are careful not to overstate the direction of this relationship: norm and attention are measured on the same forward passes and co-vary, and we do not intervene on norm independently of content. What can be said is that the specialization is not hand-designed and emerges from fine-tuning, indicating that the LLM allocates dedicated machinery to graph-conditioned positions when they carry a meaningfully different signal.

Attention entropy. We measure the Shannon entropy of the attention distribution over entity tokens per (layer, head) pair (Figure 3). In SECBFF, entropy drops sharply at layers 3 and 29–31 (matching the strongest entity heads from Figure 2), confirming peaked, confident attention decisions. The text-only model shows low entropy only in early layers (3–8), with a fundamentally different information routing strategy.

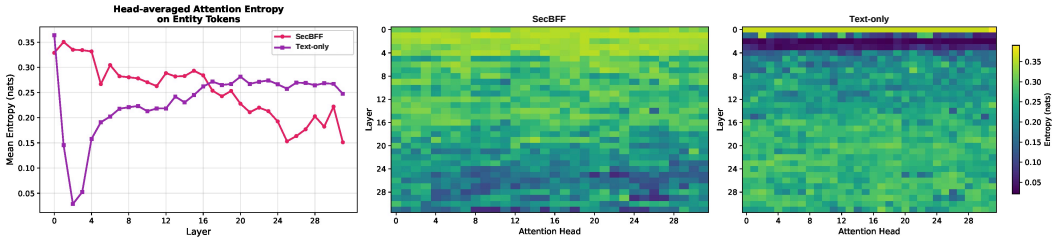

Figure 3: Attention entropy on entity tokens. **Left:** Head-averaged entropy across layers. **Center/Right:** Per-head heatmaps for SECBFF and text-only models. Lower values indicate more focused attention.

Routing and discrimination. Two further effects appear only in SECBFF. Entity self-attention stays high (0.28–0.45) across all layers, whereas in the text-only model it collapses to $\sim$0.07 after layer 6, giving a persistent subnetwork for relational reasoning between entities. The anomalous-versus-normal attention difference is bimodal, peaking at layers 0–4 and 24–31, consistent with early layers *grounding* entity representations and late layers *discriminating* against them. The Random-GFM ablation shows these effects depend on embedding *content* rather than norm, position, or coverage, which it holds fixed: detection collapses to near chance (Table 1). Appendix L gives both analyses in full with their figures, and states what the evidence does and does not establish; further analyses of GFM signal retention and embedding trajectory are given in Appendix I.

5 CONCLUSION

We presented SECBFF, a framework for temporal security detection that conditions an LLM on pretrained GFM embeddings through Topology-Injected Masked Fine-Tuning. By replacing entity-mention tokens with projected GFM embeddings at the input-embedding level, SECBFF enables the LLM to reason jointly over behavioral sequences and structural context without vocabulary expansion or architectural changes to the base model. The inherent sparsity of graph coverage ensures robustness: the model encounters uncovered entities in every training sequence, naturally learning to leverage topology when present without depending on it exclusively.

Experiments across enterprise security logs, blockchain transactions, and cybersecurity events demonstrate that SECBFF consistently outperforms text-only and graph-text baselines on classification tasks, with the largest gains in low-context settings such as zero-shot and one-event detection where topological priors are most valuable. Mechanistic analysis shows that GFM embeddings induce dedicated entity attention heads, maintain high-norm representations throughout the network, and create a persistent entity-to-entity communication subnetwork, behaviors entirely absent in text-only models. A norm-, position-, and coverage-matched ablation establishes that the learned graph geometry, rather than the act of injecting continuous vectors, is what produces them. Directions for future work are discussed in Appendix K.

REPRODUCIBILITY STATEMENT

We have made several efforts to ensure that our results are reproducible. The SECBFF framework and the Topology-Injected Masked Fine-Tuning procedure are specified in Section 3. The datasets, tasks, baselines, evaluation metrics, architectures, and hyperparameters are described in Section 4 and detailed further in the Appendix, as are the protocols for the mechanistic analysis. Two of the three datasets (Elliptic and LANL 2015) are publicly available; the enterprise security-event dataset consists of private production logs that are not publicly available, and we therefore report

full preprocessing and task-construction details in the Appendix so that the setup can be reproduced on comparable logs.

AI USE STATEMENT

In this work, we did not use generative AI tools for any of the tasks for which ICLR requires disclosure. Specifically, generative AI was not used to create synthetic datasets or to clean or reformat data, to build theoretical models or conceptual frameworks, to propose or refine hypotheses, to design the experiments or methodology or to give feedback on them, to implement the method proposed in this paper, or to interpret results. Mathematical proof writing, translation, and qualitative or thematic analysis are not applicable to this work. Among the tasks for which disclosure is recommended, we used generative AI tools for two purposes: to improve readability through editing, namely grammar and phrasing improvements to text written by the authors; and to write and edit supporting software code, such as experiment scripts, data processing utilities, and plotting and analysis code. We have reviewed all AI-assisted work. Every AI-edited sentence was checked by the authors against the intended meaning, and all AI-assisted code was read, tested, and verified for correctness by the authors before any result reported here was produced with it. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REFERENCES

Giovanni Apruzzese, Pavel Laskov, Edgardo Montes de Oca, Wissam Mallouli, Luis Brdalo Rapa, Athanasios Vasileios Grammatopoulos, and Fabio Di Franco. The role of machine learning in cybersecurity. *Digital Threats*, 4(1), March 2023. doi: 10.1145/3545574. URL https://doi.org/10.1145/3545574.

Maya Bechler-Speicher, Yoel Gottlieb, Andrey Isakov, David Abensur, Ami Tavory, Daniel Haimovich, Ido Guy, and Udi Weinsberg. Billion-scale graph foundation models, 2026. URL https://arxiv.org/abs/2602.04768.

Benjamin Bowman, Craig Laprade, Yuede Ji, and H. Howie Huang. Detecting lateral movement in enterprise computer networks with unsupervised graph AI. In *23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID)*, pp. 257–268, 2020.

Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Computing Surveys*, 41(3):1–58, 2009. doi: 10.1145/1541880.1541882.

Dawei Cheng, Yao Zou, Sheng Xiang, and Changjun Jiang. Graph neural networks for financial fraud detection: A review. *Frontiers of Computer Science*, 19(9):199609, 2025. doi: 10.1007/s11704-024-40474-y.

Xin Cheng, Xun Wang, Xingxing Zhang, Tao Ge, Si-Qing Chen, Furu Wei, Huishuai Zhang, and Dongyan Zhao. xRAG: Extreme context compression for retrieval-augmented generation with one token. In *Advances in Neural Information Processing Systems*, 2024.

David R. Cox. The regression analysis of binary sequences. *Journal of the Royal Statistical Society: Series B*, 20(2):215–242, 1958.

Min Du, Feifei Li, Guineng Zheng, and Vivek Srikumar. DeepLog: Anomaly detection and diagnosis from system logs through deep learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1285–1298, 2017.

Mohamed Amine Ferrag, Mthandazo Ndhlovu, Norbert Tihanyi, Lucas C. Cordeiro, Merouane Debbah, Thierry Lestable, and Narinderjit Singh Thandi. Revolutionizing cyber threat detection with large language models: A privacy-preserving BERT-based lightweight model for IoT/IIoT devices. *IEEE Access*, 12:23733–23750, 2024.

Wei Guan, Jian Cao, Shiyou Qian, Jianqi Gao, and Chun Ouyang. LogLLM: Log-based anomaly detection using large language models. *arXiv preprint arXiv:2411.08561*, 2024.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason E Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=Itxz7S4Ip3.

Alexander D. Kent. Comprehensive, multi-source cyber-security events. *Los Alamos National Laboratory*, 2015. https://csr.lanl.gov/data/cyber1/.

Alexander D. Kent. Cybersecurity data sources for dynamic network research. In Niall Adams and Nicholas Heard (eds.), *Dynamic Networks and Cyber-Security*, pp. 37–65. Imperial College Press, 2016.

Isaiah J. King and H. Howie Huang. Euler: Detecting network lateral movement via scalable temporal link prediction. *ACM Trans. Priv. Secur.*, 26(3), June 2023. ISSN 2471-2566. doi: 10.1145/3588771. URL https://doi.org/10.1145/3588771.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 4582–4597, 2021.

Jiawei Liu, Cheng Yang, Zhiyuan Lu, Junze Chen, Yibo Li, Mengmei Zhang, Ting Bai, Yuan Fang, Lichao Sun, Philip S. Yu, and Chuan Shi. Graph foundation models: Concepts, opportunities and challenges. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(6):5023–5044, 2025. doi: 10.1109/TPAMI.2025.3548729.

Qingyun Liu, Jack W. Stokes, Rob Mead, Tim Burrell, Ian Hellen, John Lambert, Andrey Marochko, and Weidong Cui. Latte: Large-scale lateral movement detection. In *IEEE Military Communications Conference (MILCOM)*, pp. 1–6, 2018.

Zheyuan Liu, Xiaoxin He, Yijun Tian, and Nitesh V. Chawla. Can we soft prompt LLMs for graph learning tasks? In *Companion Proceedings of the ACM Web Conference 2024 (WWW '24 Companion)*, pp. 481–484, 2024.

Haitao Mao, Zhikai Chen, Wenzhuo Tang, Jianan Zhao, Yao Ma, Tong Zhao, Neil Shah, Mikhail Galkin, and Jiliang Tang. Position: Graph foundation models are already here. In *International Conference on Machine Learning (ICML)*, 2024.

Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanezashi, Tim Kaler, Tao B. Schardl, and Charles E. Leiserson. EvolveGCN: Evolving graph convolutional networks for dynamic graphs. In *AAAI Conference on Artificial Intelligence*, 2020.

Bryan Perozzi, Bahare Fatemi, Dustin Zelle, Anton Tsitsulin, Mehran Kazemi, Rami Al-Rfou, and Jonathan Halcrow. Let your graph do the talking: Encoding structured data for LLMs. *arXiv preprint arXiv:2402.05862*, 2024.

Jiaxing Qi, Shaohan Huang, Zhongzhi Luan, Shu Yang, Carol Fung, Hailong Yang, Depei Qian, Jing Shang, Zhiwen Xiao, and Zhihui Wu. LogGPT: Exploring ChatGPT for log-based anomaly detection. In *2023 IEEE International Conference on High Performance Computing & Communications, Data Science & Systems, Smart City & Dependability in Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys)*, pp. 273–280, 2023. doi: 10.1109/HPCC-DSS-SmartCity-DependSys60770.2023.00045.

Ladislav Rampášek, Mikhail Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a general, powerful, scalable graph transformer. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

Maria Tsimpoukelli, Jacob Menick, Serkan Cabi, S. M. Ali Eslami, Oriol Vinyals, and Felix Hill. Multimodal few-shot learning with frozen language models. In *Advances in Neural Information Processing Systems*, volume 34, pp. 200–212, 2021.

Duo Wang, Yuan Zuo, Fengzhi Li, and Junjie Wu. LLMs as zero-shot graph learners: Alignment of GNN representations with LLM token embeddings. In *Advances in Neural Information Processing Systems*, 2024.

Mark Weber, Giacomo Domeniconi, Jie Chen, Daniel Karl I. Weidele, Claudio Bellei, Tom Robinson, and Charles E. Leiserson. Anti-money laundering in Bitcoin: Experimenting with graph convolutional networks for financial forensics. In *KDD Workshop on Anomaly Detection in Finance*, 2019.

Zhongjian Zhang, Xiao Wang, Mengmei Zhang, Jiarui Tan, and Chuan Shi. Toward graph-tokenizing large language models with reconstructive graph instruction tuning. In *Proceedings of the ACM Web Conference 2026*, 2026. doi: 10.1145/3774904.3792077.

Table 4: Entity representation variants. All share the same LLaMA 3 8B backbone and LoRA configuration (∼13.6M parameters). Projector parameters are additional.

Variant	Modality	Entity input	Embedding source	Proj. params
No graph, no embeddings				
No FT	Text	Plain name	—	0
Text-only	Text	Plain name	—	0
Text-graph	Text	Name + neighbors	—	0
Embedding-based (no graph structure)				
Random-GFM	Emb.	Placeholder + 128-d	Deterministic random	530K
LogLLM	Emb.	Placeholder + 768-d	BERT (text desc.)	3M
Graph-aware				
GraphPoolLinear	Graph + Text	Name + 4 prefix tok.	Aggregated GFM (linear)	530K
GraphPoolMLP	Graph + Text	Name + 4 prefix tok.	Aggregated GFM (MLP)	34M
GraphToken	Graph + Text	Name + 4 prefix tok.	2-layer GCN + attn. pool	591K
GraphPrompter	Graph + Text	Name + 4 prefix tok.	2-layer GCN + cross-attn.	102M
SecBFF (ours)	Graph + Text	Placeholder + 128-d	Pre-trained GFM	530K

A Training and Evaluation Details

We use LLaMA 3 8B as the base model with LoRA adapters of rank 16 and alpha 32 applied to the query, key, value, and output projection matrices (∼13.6M trainable parameters). Training uses AdamW with learning rate 5e-5, weight decay 0.01, betas (0.9, 0.95), epsilon 1e-8, cosine decay schedule with 50 warmup steps, gradient clipping at 1.0, and bf16 precision. For the enterprise dataset we train for 4,000 steps with batch size 4 and sequence length 1,024; for Elliptic and LANL 2015 we train for 2,000 steps with batch size 4 and sequence length 512.

The projector P_θ maps 128-dimensional GFM embeddings to the LLM hidden dimension of 4,096 via a single linear layer followed by layer normalization, introducing approximately 530,000 parameters.

We evaluate four classification methods on top of the self-supervised backbone, all trained for 1,000 steps. **Classification LoRA** trains fresh LoRA adapters with the same rank and alpha on cross-entropy loss over Yes/No answer tokens. **Prompt tuning** trains 20 soft prompt embeddings (∼80K parameters) while freezing the entire LLM. **Classification head** trains a linear layer from the 4,096-dimensional last-token hidden state to two classes (∼8K parameters) while freezing the LLM and self-supervised LoRA. **Zero-shot scoring** applies no task-specific training and computes $P(\text{Yes})/(P(\text{Yes}) + P(\text{No}))$ via softmax over the logits at the answer position.

B Baseline Descriptions

- **No FT.** The base LLaMA 3 8B model with no fine-tuning. Entities appear as plain text names.
- **Text-only.** LoRA fine-tuned on the same training data, with entities represented as plain text names. No graph information is used.
- **Text-graph.** Same text-only recipe, but the prompt is augmented with a textual description of each entity's graph neighborhood, including feature text and neighbor names.
- **Random-GFM.** Same architecture as SecBFF, but entity embeddings are deterministic random vectors instead of learned GFM embeddings. This ablation isolates the contribution of meaningful graph representations.
- **LogLLM (Guan et al., 2024).** Uses BERT embeddings computed from textual entity descriptions, with 3M projector parameters.
- **GraphPoolLinear.** Aggregates GFM node embeddings from entity subgraphs via $K{=}4$ separate linear projections, producing four prefix tokens prepended to the LLM input alongside the entity name, with 530K projector parameters.

- **GraphPoolMLP.** Same as GraphPoolLinear but uses an MLP for aggregation, with 34M projector parameters.
- **GraphToken (Perozzi et al., 2024).** A 2-layer GCN encodes entity subgraphs, attention-weighted pooling produces $K{=}4$ token vectors, and a linear layer projects them into the LLM embedding space, with 591K projector parameters.
- **GraphPrompter (Liu et al., 2024).** A 2-layer GCN encodes entity subgraphs, and learnable prompt tokens cross-attend to the GNN node embeddings via multi-head attention with a feed-forward network, producing $K{=}4$ prefix vectors, with 102M projector parameters.

C Enterprise Anomaly Injection Protocol

The enterprise dataset is proprietary and its anomalies are synthetically injected. We give the injection procedure in full, both for transparency about what the labels mean and because the procedure bears directly on whether the benchmark could favor graph-aware methods by construction.

Per-sequence procedure. Each user's event sequence is processed independently.

1. **Context/suffix split.** Events are split 50/50. The first $\lfloor n/2 \rfloor$ events form the context and are never modified, always carrying label 0; only suffix events are injection candidates. Sequences with fewer than four events are left untouched and are entirely benign.
2. **Injection sampling.** Each suffix event is independently made anomalous with probability 0.15. The realized anomaly fraction is therefore $\approx 0.15 \times 0.5 = 7.5\%$ of all events; the exact realized rate is logged per run.
3. **Anomaly type** is chosen uniformly among three, each a proxy for a real attack behavior:
   - **novel_entity**: the destination is replaced by an entity ID never seen in *that user's history* (a random 8-digit ID with type drawn from {UnknownBiz, SuspiciousAcct, ForeignMerchant}). Proxies first-contact or lateral movement to an unfamiliar resource.
   - **temporal_burst**: the event's timestamp is overwritten with the previous event's timestamp, producing a rapid-fire duplicate-timestamp burst. Proxies automated or scripted activity.
   - **type_shift**: the entity type is swapped to a rare or suspicious type among five possible types while the ID is kept. Proxies access to a resource of an anomalous class.
4. **Labels.** A per-event binary label array is emitted: 0 for all context events, 0/1 for suffix events, with 1 marking an injected anomaly. Evaluation is at the event level from these labels.

Independence from graph topology. A natural concern is that these anomaly types are not independent of the entity graph, since "novel entity access" in particular sounds close to graph-definitional. Were that so, the benchmark would mechanically advantage graph-aware methods on this dataset. Three properties of the construction rule this out.

First, all three anomaly types are defined purely in *feature and event space*, with reference only to the user's own event stream: novelty is defined against that user's prior history in the log, burstiness against inter-event timing, and type shift against the entity's declared type. The injection code never queries the graph. In particular, "novel" means unseen in the user's temporal event stream, not absent from the graph; an injected entity may well be graph-covered, and a long-familiar entity may not be.

Second, injected entity IDs are sampled uniformly at random, independent of degree, centrality, community membership, or coverage status. Anomaly labels are therefore uncorrelated with topology by construction, and every method, graph-aware or not, receives an identical feature set and identical labels.

Third, and most directly, the enterprise GFM is pre-trained on a social and financial interaction network that is *completely disjoint* from the downstream evaluation graph, so it cannot encode the injected entities at all. Finally, the same qualitative gains appear on Elliptic and LANL, where

Table 5: AUC-PR on Elliptic Bitcoin, complementing the AUC-ROC results of Table 2. The illicit base rate is 5.9%, which is the score a random detector would obtain. Mean ± standard deviation over three random seeds.

	Method	Sequence AUC-PR	Real-Time AUC-PR	Zero-Shot RT AUC-PR	One-Event ZS RT AUC-PR
Abl.	No FT	$0.09_{\pm 0.03}$	$0.12_{\pm 0.03}$	$0.05_{\pm 0.02}$	$0.05_{\pm 0.02}$
	Random-GFM	$0.17_{\pm 0.03}$	$0.15_{\pm 0.03}$	$0.06_{\pm 0.02}$	$0.07_{\pm 0.02}$
Text	Text-only	$0.42_{\pm 0.02}$	$0.40_{\pm 0.02}$	$0.08_{\pm 0.02}$	$0.08_{\pm 0.02}$
	LogLLM	$0.41_{\pm 0.03}$	$0.44_{\pm 0.02}$	$0.08_{\pm 0.02}$	$0.08_{\pm 0.02}$
	Text-graph	$0.44_{\pm 0.02}$	$0.40_{\pm 0.02}$	$0.06_{\pm 0.02}$	$0.09_{\pm 0.02}$
Graph+Text	GraphPoolMLP	$0.16_{\pm 0.03}$	$0.11_{\pm 0.03}$	$0.07_{\pm 0.02}$	$0.07_{\pm 0.02}$
	GraphPoolLinear	$0.14_{\pm 0.03}$	$0.16_{\pm 0.03}$	$0.07_{\pm 0.02}$	$0.06_{\pm 0.02}$
	GraphToken	$0.39_{\pm 0.02}$	$0.46_{\pm 0.02}$	$0.09_{\pm 0.02}$	$0.08_{\pm 0.02}$
	GraphPrompter	$0.42_{\pm 0.02}$	$0.41_{\pm 0.02}$	$0.08_{\pm 0.02}$	$0.07_{\pm 0.02}$
	SecBFF	$\mathbf{0.51}_{\pm \mathbf{0.02}}$	$\mathbf{0.49}_{\pm \mathbf{0.02}}$	$\mathbf{0.13}_{\pm \mathbf{0.02}}$	$\mathbf{0.14}_{\pm \mathbf{0.02}}$

anomalies are real, LANL provides ground-truth red-team labels, and no injection procedure exists. The advantage of SECBFF is therefore not an artifact of how the enterprise anomalies were produced.

D PRECISION–RECALL RESULTS FOR ELLIPTIC AND LANL

Elliptic and LANL are highly imbalanced (5.9% illicit and 2.8% compromised respectively), and AUC-ROC can mask weak precision under such imbalance. We therefore report AUC-PR for every method and setting on both datasets in Tables 5 and 6, alongside the AUC-ROC results of Tables 2 and 3.

The ranking is unchanged: SECBFF is the best method in all eight dataset/setting combinations. More importantly, the margins are *wider* under AUC-PR than under AUC-ROC, which is the opposite of what the imbalance concern would predict. On LANL sequence classification SECBFF improves over the text-only baseline by 111% in AUC-PR (0.38 vs. 0.18) against 18.8% in AUC-ROC, and over the strongest baseline, GraphPrompter, by 65%. On Elliptic the corresponding sequence-level gain over text-only is 21% in AUC-PR against 7.4% in AUC-ROC. Relative to the base rate a random detector would achieve, SECBFF reaches $8.6\times$ on Elliptic and $13.6\times$ on LANL. The low-context settings show the same amplification: on Elliptic one-event zero-shot, SECBFF attains 0.14 AUC-PR against 0.09 for the best competing method, a 56% margin.

The reason is that AUC-ROC rewards correct ordering across the whole score range, including the large benign majority that every method ranks correctly, whereas AUC-PR is dominated by the top of the ranking, which is where an analyst actually looks. Topological priors help most precisely there: they let the model distinguish among events that are textually similar but structurally different, which is a high-scoring-region distinction. We note one limitation of this evidence. AUC-PR is a threshold-free summary, and we do not report the fully operational metrics an alert pipeline would be tuned on, such as precision at a fixed daily alert budget, recall at a fixed false-positive rate, or false positives per day. Those depend on deployment-specific triage capacity that our datasets do not specify.

E COMPARISON TO NON-LLM DETECTORS

Setup. The baselines in the main text isolate *how* to inject topology into an LLM, holding the LLM fixed. A separate question is whether an LLM should be the predictor at all, given the substantial literature on graph-based and log-based security detection. We therefore evaluate seven established non-LLM detectors: the tabular classifiers Random Forest (Breiman, 2001) and Logistic Regres-

Table 6: AUC-PR on LANL 2015, complementing the AUC-ROC results of Table 3. The compromised base rate is 2.8%, which is the score a random detector would obtain. Mean ± standard deviation over three random seeds.

	Method	Sequence AUC-PR	Real-Time AUC-PR	Zero-Shot RT AUC-PR	One-Event ZS RT AUC-PR
Abl.	No FT	$0.05_{\pm 0.02}$	$0.06_{\pm 0.02}$	$0.03_{\pm 0.01}$	$0.03_{\pm 0.01}$
	Random-GFM	$0.13_{\pm 0.03}$	$0.20_{\pm 0.03}$	$0.04_{\pm 0.01}$	$0.04_{\pm 0.01}$
Text	Text-only	$0.18_{\pm 0.03}$	$0.22_{\pm 0.03}$	$0.04_{\pm 0.01}$	$0.04_{\pm 0.01}$
	LogLLM	$0.14_{\pm 0.03}$	$0.16_{\pm 0.03}$	$0.05_{\pm 0.01}$	$0.05_{\pm 0.01}$
	Text-graph	$0.17_{\pm 0.03}$	$0.21_{\pm 0.03}$	$0.03_{\pm 0.01}$	$0.04_{\pm 0.01}$
Graph+Text	GraphPoolMLP	$0.06_{\pm 0.02}$	$0.05_{\pm 0.02}$	$0.04_{\pm 0.01}$	$0.05_{\pm 0.01}$
	GraphPoolLinear	$0.05_{\pm 0.02}$	$0.06_{\pm 0.02}$	$0.05_{\pm 0.01}$	$0.05_{\pm 0.01}$
	GraphToken	$0.19_{\pm 0.03}$	$0.21_{\pm 0.03}$	$0.05_{\pm 0.01}$	$0.05_{\pm 0.01}$
	GraphPrompter	$0.23_{\pm 0.03}$	$0.23_{\pm 0.03}$	$0.04_{\pm 0.01}$	$0.04_{\pm 0.01}$
	SecBFF	$\mathbf{0.38}_{\pm \mathbf{0.02}}$	$\mathbf{0.34}_{\pm \mathbf{0.02}}$	$\mathbf{0.08}_{\pm \mathbf{0.02}}$	$\mathbf{0.09}_{\pm \mathbf{0.02}}$

Table 7: Non-LLM security and graph detectors evaluated on the same event streams, temporal splits, and sequence-level labels as Tables 2 and 3. Protocol details are given in Appendix E.

Method	Dataset	Sequence AUC-ROC
Random Forest	Elliptic	0.51
Logistic Regression	Elliptic	0.51
GCN / GraphGPS	Elliptic	0.50
EvolveGCN-O	Elliptic	0.51
Bowman et al.	LANL	0.51
DeepLog	LANL	0.54
Euler	LANL	0.50

sion (Cox, 1958); the static graph models GCN (Kipf & Welling, 2017) and GraphGPS (Rampášek et al., 2022); the temporal graph models EvolveGCN-O (Pareja et al., 2020) and Euler (King & Huang, 2023); the unsupervised lateral-movement detector of Bowman et al. (2020); and the log-sequence anomaly detector DeepLog (Du et al., 2017); results are reported in Table 7. All seven use the same event streams, temporal splits, and sequence-level labels as the LLM methods, so the numbers are directly comparable to Tables 2 and 3: each is trained on the training split and evaluated on the held-out split with no access to test-time labels, per-event scores are aggregated to the sequence level, and we score the *full* stream rather than filtering out events whose entities lack graph coverage, which would evaluate an easier task than the one SECBFF solves. For the graph-based detectors, entities absent from the pre-training graph have no learned representation; we assign them a zero-vector embedding and let the model score them as usual rather than abstaining.

Results. All seven perform at or near chance on the full stream (0.50–0.54 AUC-ROC), far below both the text-only LLM baseline and SECBFF, which reaches 0.796 on Elliptic and 0.859 on LANL under the same protocol. These methods are handicapped *by construction* in this regime, and the result should be read as a property of the setting rather than as a claim about their quality. Coverage is the dominant factor: with embeddings available for only 15–25% of entities, a graph-only detector has no representation for the majority of the events it must score, and no fill-in convention repairs this, since abstaining, mean imputation, and a degree prior differ only in how that uninformative majority is labelled. The real-time and zero-shot settings compound the problem by requiring scores for entities that may not yet appear in any graph build, while the tabular and log-sequence detectors discard the entity semantics that the LLM reads directly from text. The Elliptic numbers additionally reflect the evaluation window: published results there are node-level over the full la-

belled set, whereas we evaluate sequence-level detection over timesteps 35–49, which straddles the dark-market shutdown at timestep 43 after which the illicit-class performance of every supervised method evaluated by Weber et al. (2019) collapses, so near-chance performance for a feature-based classifier in this window is the expected result rather than one in tension with the original paper. DeepLog is the strongest of the seven at 0.54, consistent with this account, since it is the only one that models the temporal stream directly without depending on graph coverage. The conclusion is therefore about problem framing rather than ranking: under partial coverage the graph cannot serve as the predictor, because it does not see most of the data, and can only serve as *side information* for a predictor that does, which is precisely the role topology plays in SECBFF.

F DATASET CONSTRUCTION DETAILS

LANL 2015. We construct a temporal interaction graph with 449,351 nodes (computer-hourly-bucket pairs across 13,850 unique computers and 566 hourly buckets), 290,852 directed edges, and 21 features per node encoding time bucket, authentication orientation and type counts, success and failure rates, flow statistics, process start and stop counts, and DNS query volume, all log-transformed. We use a temporal split with hourly buckets 0 through 4 for graph pre-training (30,788 nodes, 142,978 edges) and buckets 5 onward for evaluation (418,563 nodes, 147,874 edges, 2.2% compromised). Sequences are generated using the same ego-graph approach as Elliptic with three hops and up to 100 neighbors, producing 26,059 sequences with 717 compromised (2.8% anomaly rate). The graph transformer is pre-trained in self-supervised mode on the early-bucket subgraph.

Elliptic Bitcoin. Each node has 166 features consisting of 94 local transaction features and 72 aggregated one-hop neighbor features. Event sequences are constructed via breadth-first ego-graph expansion with three hops, a maximum of 100 neighbors, and a minimum of five edges per sequence. The graph transformer is pre-trained locally on the training-split subgraph. Because the Elliptic graph is small relative to the scale GraphBFF targets, link prediction alone is a weak pre-training signal, so we use a multi-task objective that adds a *self-supervised* node-classification head predicting held-out node *features* (not labels), with equal loss weighting. No illicit/licit transaction labels are used at any point during GFM pre-training; all label information enters only through the downstream LLM classification adapter. The zero-shot settings are therefore genuinely label-free with respect to topology.

G PROMPT TEMPLATES

Three prompt templates are appended after the event sequence depending on the evaluation task. Scoring uses $P(\text{Yes})/(P(\text{Yes}) + P(\text{No}))$ via softmax over the logits at the answer position, where the target tokens are `" Yes"` and `" No"` (with leading space).

- **Per-event classification:** `"The user above just performed the following action: {event}. Is this action anomalous for this user? Answer:"`
- **Sequence classification:** `"Based on the user's full activity above, is this user's behavior anomalous? Answer:"`
- **Last-event classification:** `"Based on the user's full activity above, is the user's most recent action anomalous? Answer:"`

H ROBUSTNESS TO TOPOLOGICAL COVERAGE

This appendix gives the full coverage study summarized in Section 4.6. Using Elliptic, we subsample the set of entities for which a GFM embedding is exposed to the model, sweeping coverage from 3% to 25% while holding the data, the graph model, and all hyperparameters fixed.

Table 8: Controlled coverage sweep on Elliptic. The fraction of entities with an available GFM embedding is varied from 3% to 25%; the 15% row corresponds to the natural coverage used in Table 2. Mean ± standard deviation over three seeds.

Coverage	Sequence AUC-ROC	Real-Time AUC-ROC
3%	$0.741_{\pm 0.003}$	$0.732_{\pm 0.005}$
5%	$0.750_{\pm 0.005}$	$0.738_{\pm 0.001}$
10%	$0.774_{\pm 0.008}$	$0.785_{\pm 0.009}$
15%	$0.796_{\pm 0.009}$	$0.792_{\pm 0.005}$
20%	$0.821_{\pm 0.001}$	$0.799_{\pm 0.002}$
25%	$\mathbf{0.824_{\pm 0.001}}$	$\mathbf{0.815_{\pm 0.002}}$

Table 9: Train/inference coverage shift on Elliptic. Each cell reports the model trained at 3% coverage / the model trained at 25% coverage, both evaluated at the inference coverage given in the first column. Mean ± standard deviation over three seeds.

Inference coverage	Sequence AUC-ROC train 3% / train 25%	Real-Time AUC-ROC train 3% / train 25%
5%	$0.751_{\pm 0.001}$ / $0.735_{\pm 0.004}$	$0.752_{\pm 0.003}$ / $0.762_{\pm 0.001}$
10%	$0.750_{\pm 0.002}$ / $0.745_{\pm 0.001}$	$0.748_{\pm 0.006}$ / $0.722_{\pm 0.001}$
15%	$0.764_{\pm 0.001}$ / $0.761_{\pm 0.004}$	$0.775_{\pm 0.001}$ / $0.742_{\pm 0.009}$

COVERAGE SWEEP

Table 8 shows that SECBFF degrades gracefully as topology becomes scarce. Performance falls monotonically but gently, from 0.824 sequence AUC-ROC at 25% coverage to 0.741 at 3% coverage, a drop of 0.083 for an eight-fold reduction in available structure. The relevant reference point is the text-only baseline, which reaches 0.741 sequence and 0.735 real-time AUC-ROC on Elliptic (Table 2). At 3% coverage SECBFF scores 0.741 and 0.732, within one standard deviation of text-only on both metrics. In other words, the text-only model acts as an empirical *floor* for SECBFF: topology too sparse to help does not measurably hurt, and every additional covered entity contributes monotonically thereafter (both metrics increase monotonically across all six coverage levels). This is the behavior the masked-injection formulation is designed to produce, since the model sees uncovered entities in every training sequence and never learns to require a topology token. It also indicates that the 15% coverage of our main Elliptic results is not a favorable operating point chosen after the fact; it sits in the middle of a smooth curve.

COVERAGE SHIFT BETWEEN TRAINING AND INFERENCE

Deployed systems rarely see the same coverage at inference that they saw during training, since graph builds lag behind the event stream. Table 9 tests this mismatch directly by training at the two extremes of the sweep and evaluating at intermediate coverage. Degradation is modest overall, across the twelve mismatched cells the median loss relative to the coverage-matched model of Table 8 is 0.027 AUC-ROC, but it is markedly asymmetric in the direction of the shift. Training at low coverage transfers better than training at high coverage in five of the six comparisons, and the distinction is sharper than the averages suggest: *every* configuration trained at 3% coverage matches or exceeds the text-only baseline at every inference coverage, whereas training at 25% and evaluating below it produces the only two cells that fall beneath text-only (0.735 vs. 0.741 sequence at 5% coverage; 0.722 vs. 0.735 real-time at 10%) and the largest single degradation (0.722 vs. 0.785 coverage-matched, a loss of 0.063).

This asymmetry is the expected consequence of the masked-injection formulation. A model trained at 25% coverage encounters topology tokens often enough to become partially reliant on them and pays a real penalty when they are withheld; a model trained at 3% coverage has already learned to treat topology as an optional signal and loses little when it becomes scarcer, while still benefiting when more arrives. The practical implication is a concrete deployment recommendation: train at or below the coverage expected at inference, never above it. Robustness to sparse topology therefore does not transfer automatically to topology that is *sparser than that seen during training*.

I Additional Analysis

GFM signal retention. We measure the cosine similarity between the projected GFM embedding injected at layer 0 and the entity hidden state at each subsequent layer (Figure 4). The signal decays smoothly from $\sim$1.0 at layer 0 to $\sim$0.73 at layer 31, with $>$0.95 similarity retained through layer 14. The GFM signal is never washed out, allowing all layers to leverage structural information.

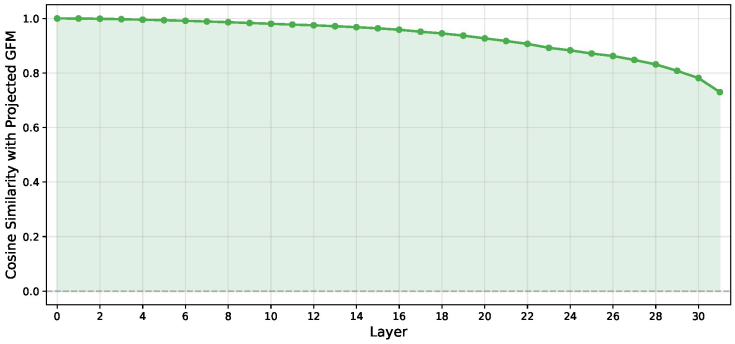

Figure 4: GFM signal retention. Cosine similarity between the projected GFM embedding and entity hidden states at each layer, averaged over 91 samples.

Embedding trajectory. We extract entity hidden states at layers $\{0, 4, 8, 12, 16, 20, 24, 28, 31\}$ and project them to 2D via PCA. In SecBFF (Figure 5), entity embeddings maintain high variance throughout (PC1 explains $\sim$60% consistently) and form structured clusters by layer 31, with entities separating according to their behavioral profiles in the interaction graph. In the text-only model (Figure 6), embeddings start nearly one-dimensional (PC1=98.5% at layer 0), diversify through mid-layers (PC1 drops to $\sim$39% by layer 20), and reconcentrate at the output (PC1=67.3% at layer 31).

J Datasets Information

J.1 LANL 2015 Comprehensive, Multi-Source Cyber-Security Events Dataset

We use the LANL 2015 dataset Kent (2015), comprising 58 consecutive days of de-identified event data collected from five sources within Los Alamos National Laboratory's enterprise network. Crucially, this dataset includes red team ground truth labels, enabling supervised evaluation of anomaly detection methods.

J.1.1 Overall Scale

Table 10 summarizes the overall scale of the dataset.

Table 10: LANL 2015 dataset overview.

Statistic	**Value**
Total temporal events	1,648,274,558
Observation period	58 days
Unique device nodes	17,684
Unique users	88,916
Unique processes	62,974
Data sources	5 (Auth, Flows, DNS, Process, Red Team)

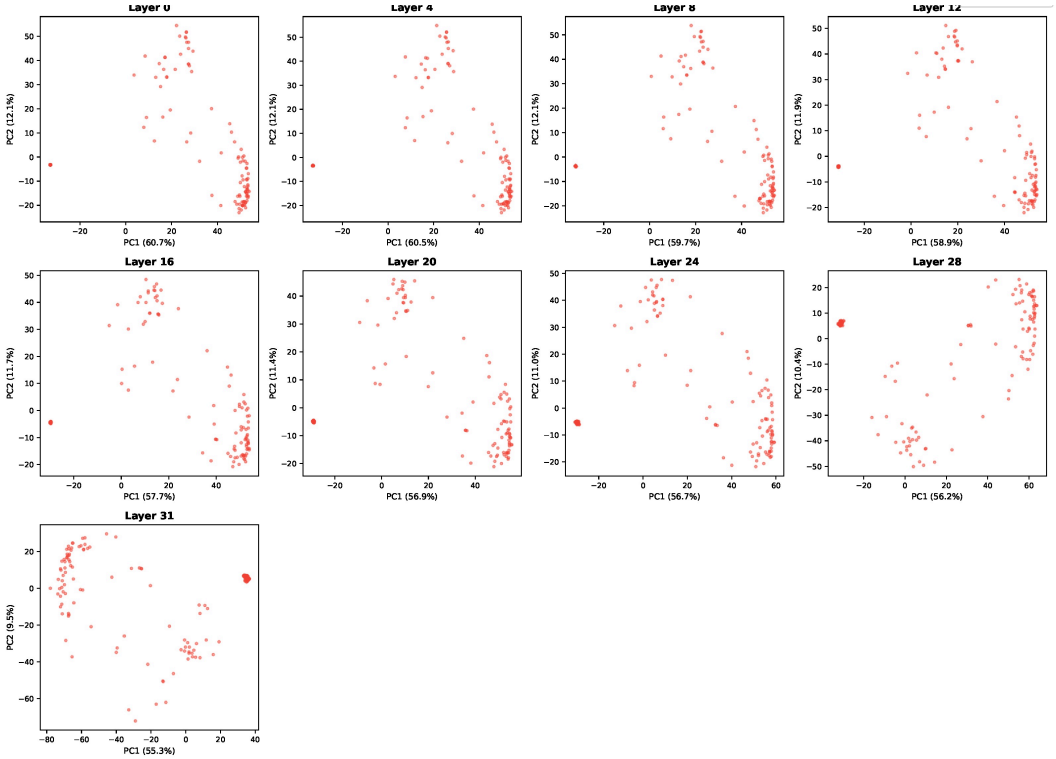

Figure 5: Entity token embedding trajectory across layers (SECBFF). PCA projections at 9 layers show high-variance, structured representations.

J.1.2 RED TEAM LABELS (GROUND TRUTH)

The dataset includes 749 known compromise events representing lateral movement by a red team. Each label is a tuple (*Time*, *User*, *SrcComputer*, *DstComputer*) extracted from the authentication data, enabling direct join with auth events for supervised evaluation.

Table 11 reports the red team label statistics.

Table 11: Red team label statistics.

Statistic	**Value**
Total compromise events	749
Unique compromised users	104
Unique source (attacker) computers	4
Unique target computers	301
Label time span	150,885 – 2,557,047 (seconds)

The label granularity is **per-edge**: each red team event marks a specific authentication event (user lateral movement from one computer to another) as malicious. The extreme class imbalance, 749 malicious events out of over 1 billion authentication events, reflects the realistic needle-in-a-haystack nature of the detection task.

J.1.3 AUTHENTICATION EVENTS

Table 12 reports the authentication event statistics.

Edge features (7). Each authentication event carries: *SrcUser*, *DstUser*, *SrcComputer*, *DstComputer*, *AuthType*, *LogonType*, *AuthOrientation*, and *Status*.

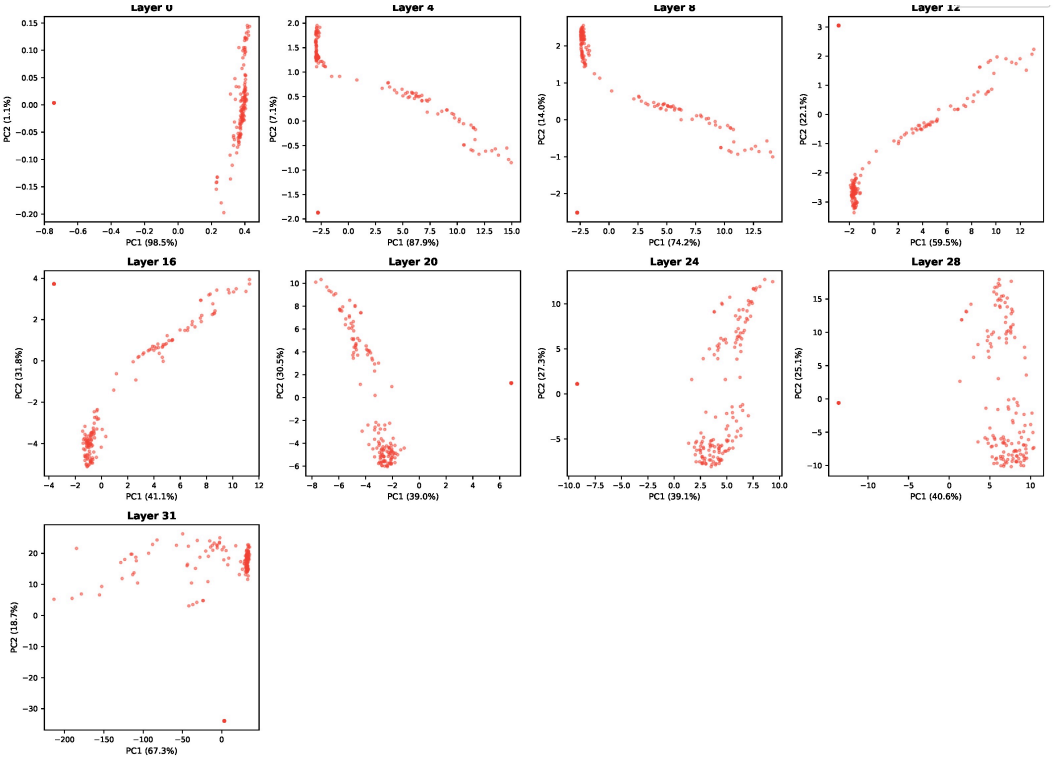

Figure 6: Entity token embedding trajectory across layers (text-only baseline). Representations start one-dimensional, diversify in mid-layers, and compress at the output.

Table 12: Authentication event statistics.

Statistic	Value
Total events	1,051,430,459
Unique nodes (computers)	17,666
Unique users	80,553
Unique static edges	419,744
Avg static degree	47.5
Avg temporal degree (events/node)	59,517.2
Median temporal degree	27,965
Max temporal degree	176,626,899

Authentication type breakdown. Table 13 gives the breakdown by authentication type. Table 14 gives the breakdown by logon type.

Table 13: Authentication types.

Auth Type	Count	Fraction
Unknown (?)	583,350,819	55.5%
Kerberos	362,830,725	34.5%
Negotiate	55,047,638	5.2%
NTLM	49,341,300	4.7%
Other	859,977	0.1%

Logon type breakdown.

Table 14: Logon types.

Logon Type	Count	Fraction
Network	845,706,092	80.4%
Unknown (?)	147,250,039	14.0%
Service	48,894,653	4.7%
Unlock	3,969,891	0.4%
Interactive	2,052,831	0.2%
Batch	1,789,374	0.2%
NewCredentials	1,113,888	0.1%
NetworkCleartext	391,055	<0.1%
RemoteInteractive	153,611	<0.1%
CachedInteractive	109,025	<0.1%

Status breakdown. 98.8% of authentication events are successful (1,038,590,151), with 1.2% failures (12,840,308).

J.1.4 Network Flow Data

Table 15 reports the network flow statistics.

Table 16 gives the breakdown by protocol.

Table 15: Network flow statistics.

Statistic	Value
Total events	129,977,412
Unique nodes	12,027
Unique static edges	162,157
Avg static degree	27.0
Avg temporal degree	10,807.1
Median temporal degree	5,243
Max temporal degree	17,441,599

Protocol breakdown. Table 17 reports the DNS lookup statistics.

Table 16: Network flow protocols.

Protocol	Type	Events	Fraction
6	TCP	116,550,579	89.7%
17	UDP	10,104,702	7.8%
1	ICMP	3,321,625	2.6%
41	IPv6	506	<0.1%

Edge features (7). Timestamp, duration, source port, destination port, protocol, packet count, byte count.

J.1.5 DNS Lookup Events

Table 18 reports the process event statistics.

Each DNS event records a (*Time*, *SrcComputer*, *ResolvedComputer*) tuple.

Table 17: DNS lookup statistics.

Statistic	Value
Total events	40,821,591
Unique nodes	15,296
Unique static edges	425,025
Avg static degree	55.6

Table 18: Process event statistics.

Statistic	Value
Total events	426,045,096
Unique hosts	11,960
Unique users	33,577
Unique process names	62,974
Start events	335,842,571 (78.8%)
End events	90,202,525 (21.2%)

J.1.6 PROCESS EVENTS

Each process event records (*Time*, *User*, *Computer*, *ProcessName*, *Start/End*). These are node-local events providing behavioral context per host.

J.1.7 MULTI-RELATIONAL GRAPH VIEW

The four data sources share a common `Computer` namespace, with strong overlap: 11,764 computers appear in all three edge-producing sources (auth, flows, DNS).

Table 19 summarizes the properties of the combined multi-relational temporal graph.

Table 19: Combined multi-relational temporal graph properties (LANL 2015).

Property	Value
Node types	1 (Computer)
Edge relation types	13 (4 auth types + 4 flow protocols + DNS + 3 auth orientations)
Total temporal edges	$\sim$1.2B (auth + flows + DNS)
Total node-level events	$\sim$426M (process start/end)
Unique static edges (all sources)	740,031
Unique nodes	17,684
Labels	749 red team events (per-edge, lateral movement)
Class imbalance	$\sim$1 : 1,400,000

Detection task. The primary task is **edge-level binary classification**: given a temporal authentication event (user, source computer $\rightarrow$ destination computer), predict whether it constitutes a red team lateral movement. The extreme class imbalance (749 positive out of $\sim$1.05B authentication events, ratio $\sim$1:1.4M) makes this a challenging anomaly detection problem. Node-level features from process events and contextual edges from DNS and netflow provide auxiliary signals for graph-based detection approaches.

Node overlap between sources. Table 20 quantifies the node overlap between sources.

J.2 GRAPH PRE-TRAINING

All datasets use the same graph transformer architecture for GFM pre-training: three layers of PyG TransformerConv with four attention heads, hidden dimension 128, ReLU activation, layer normalization, and dropout rate 0.1. A link prediction head concatenates source and target embeddings

Table 20: Node overlap between data sources.

Source Intersection	Shared Nodes
Auth ∩ Flows	12,012
Auth ∩ DNS	15,278
Flows ∩ DNS	11,779
Auth ∩ Flows ∩ DNS	11,764

and maps through a two-layer MLP (256 to 128 to 1) with binary cross-entropy loss. Edges are randomly split 80/20 with disjoint training ratio 0.2 and negative sampling ratio 1.0, and the graph is made undirected before training. We train for 200 epochs using AdamW with learning rate 1e-3, weight decay 0.01, cosine annealing schedule with minimum learning rate 1e-4, and gradient clipping at 1.0. For Elliptic we additionally add a *self-supervised* node-feature prediction head (Linear 128 to 128) trained jointly with the link prediction objective; no illicit/licit transaction labels are used at any point in GFM pre-training. For the enterprise dataset, entity embeddings come from an existing production GFM and no local pre-training is performed.

K Future Work

Generality across graph models. All three of our datasets draw on the same graph model family and the same 128-dimensional embeddings. We treat the graph model as a frozen black box by design, and our contribution concerns how to integrate such embeddings rather than how to produce them, which makes the sensitivity of TIM-FT to embedding dimensionality, encoder architecture, and pre-training objective an open and readily testable question. Establishing that the gains transfer across graph models, including much smaller and much larger embedding spaces, is in our view the most valuable next experiment, and a positive result would considerably broaden the applicability of the approach.

Mechanism across domains. The mechanistic analysis of Section 4.7 is conducted on the enterprise dataset. Whether the same internal structure (entity heads, sustained norms, and the entity-to-entity subnetwork) emerges on Elliptic and LANL would indicate whether we have identified a general consequence of topology injection or a property of one setting. A complementary and sharper experiment is interventional rather than observational: ablating the identified entity heads and measuring the resulting change in detection would establish whether those heads are the causal pathway or a correlate of it, which our present evidence cannot distinguish.

Coverage extremes and coverage dynamics. We characterize behavior between 3% and 25% coverage. The regime below 3% is untested, though the trend suggests convergence to text-only performance rather than degradation past it. More interesting is the dynamic case: real deployments do not hold coverage fixed but accumulate it as graph builds catch up with the event stream. Our coverage-shift results (Table 9) show that the direction of that drift matters, which suggests training procedures that anticipate growing coverage, such as scheduling coverage during fine-tuning rather than sampling it at a fixed rate.

Operational evaluation. We report threshold-free metrics. Moving to the metrics an alert pipeline is actually tuned on (precision at a fixed daily alert budget, recall at a fixed false-positive rate, and false positives per analyst-day) requires a deployment context that specifies triage capacity, and would test whether the AUC-PR gains reported here translate into a reduced analyst workload. Related is the question of disparate false-positive rates across user populations, which we do not measure and which matters for any monitoring system acting on real people.

Beyond security. Finally, while we focus on security applications, where sparse topological coverage and rapidly evolving event streams are particularly common, the underlying problem, conditioning a language model on structured side information available for only a subset of the entities it reasons about, is not specific to this domain. Extending TIM-FT to settings such as clinical event sequences, recommendation logs, or supply-chain records is a promising direction.

L MECHANISTIC ANALYSIS: FULL ARGUMENT AND SCOPE

This appendix continues the mechanistic analysis of Section 4.7, which established entity norm dynamics and dedicated entity heads. Throughout we distinguish two kinds of effect, because they carry different evidential weight: some properties of the injected representations are *imposed* by the architecture and would hold for any injected vector, while others are *learned*, depending on the content of the GFM embedding and disappearing when that content is destroyed. Only the second kind can explain a difference in detection performance, and we use the Random-GFM ablation to separate them.

Entity-to-entity communication. Beyond attending *to* entities, the model also routes attention *between* them. Figure 7 measures the fraction of attention that entity tokens direct to other entity tokens. In SECBFF, entity self-attention is consistently high (0.28–0.45) across all layers, rising to $\sim$0.45 by layers 24–31. In the text-only model, self-attention collapses to $\sim$0.07 after layer 6. This persistent entity communication subnetwork is the structural precondition for relational reasoning between entities within a sequence, allowing the model to compare, for example, whether a user's current access target is consistent with their historical interaction partners. It is absent in the text-only model, where entities are isolated tokens with no preferential routing.

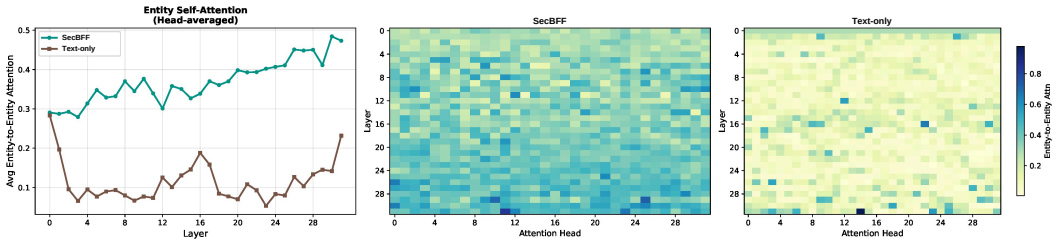

Figure 7: Entity-to-entity attention across layers. SECBFF maintains persistent inter-entity communication ($\sim$0.45 in late layers); text-only collapses to $\sim$0.07 after layer 6.

Two-phase anomaly detection. Figure 8 compares entity attention between anomalous and normal sequences. The attention difference follows a bimodal pattern: an early peak at layers 0–4 ($\sim$0.14), a mid-layer trough ($\sim$0.02–0.04), and a late peak at layers 24–31 ($\sim$0.10–0.13). This is consistent with the model operating in two phases: early layers *ground* entity representations by integrating graph context into the sequence, while late layers *discriminate* by comparing observed behavior against the grounded entity profiles, with the mid-layer trough corresponding to contextual integration.

This contrast is *within-model*: it is evaluated over the same model, the same entities, the same injection positions, and the same norms, so the surviving attention difference cannot be attributed to norm, position, or coverage. It also offers an explanation for the zero-shot results, since if grounding is already performed by the self-supervised backbone, anomaly-relevant information can reach the output layers without task-specific supervision.

ISOLATING LEARNED GEOMETRY FROM NORM, POSITION, AND COVERAGE

The observations above are between-model comparisons, and a natural objection is that they reflect trivial consequences of injecting any high-norm continuous vector rather than anything about graph structure. The Random-GFM ablation addresses this directly, and it is an intervention rather than a correlation: it holds the architecture, the projector, the training data, and the optimization fixed, and changes only the *content* of the injected vectors.

The control is tight on exactly the three factors at issue. Norm is matched, because the projector's layer normalization fixes the injected-token norm to $\sqrt{d_\ell} = 64$ irrespective of its input. Position is matched, because injection occurs at the same entity mentions. Coverage is matched, because the same entity set is covered. If high norms at entity positions were sufficient to produce the observed behavior, Random-GFM would reproduce it.

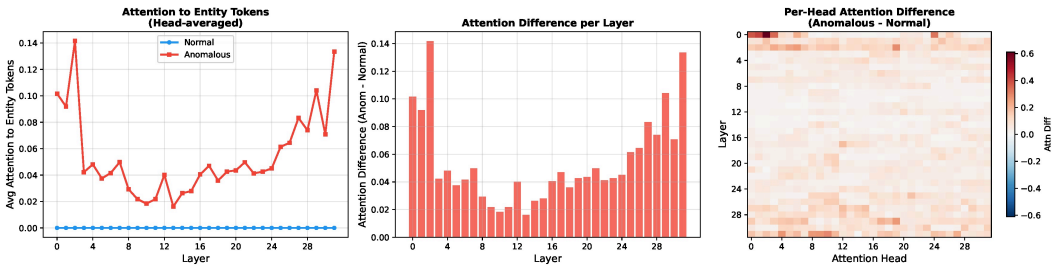

Figure 8: Anomaly-driven attention flow. The bimodal pattern is consistent with a two-phase mechanism: early entity grounding followed by late anomaly discrimination. This is a within-model contrast, with entities, positions, and norms held constant.

It does not. Detection collapses from 0.992 to 0.371 on sequence classification and from 0.872 to 0.545 on zero-shot real-time (Table 1), and the same pattern holds on Elliptic (0.796 to 0.598) and LANL (0.859 to 0.673). The magnitude is worth noting: at 0.371, Random-GFM performs *worse than no fine-tuning at all* (0.711) and far below text-only (0.949). Injecting norm-matched but uninformative vectors at entity positions is therefore not merely unhelpful but actively harmful, which is difficult to reconcile with any account in which the benefit comes from norm, position, or added capacity. Since these three factors are held constant across the intervention, the learned geometry of the GFM embedding is necessary for the effect.

Summary and scope

Norm and position supply the *capacity* to attend: sustained high-norm representations at entity positions make it possible for heads to specialize there, and this capacity is architecturally available to any injected vector. Learned graph geometry supplies the *discriminative signal*, determining whether that capacity is used to any purpose. The entity-to-entity subnetwork is the structure through which the signal is routed, and the two-phase grounding-then-discrimination pattern is the function it implements; none of these behaviors appear in the text-only model. This also explains why prefix-based fusion underperforms in-place replacement: prepending graph tokens places structural information away from the entity mention positions where relational comparison occurs, and creates no positional norm contrast at those positions.

We are explicit about the limits of this account. The Random-GFM intervention establishes that learned geometry is *necessary* for the performance gain, with norm, position, and coverage held fixed, and the within-model contrast of Figure 8 establishes that the two-phase pattern is anomaly-driven rather than an artifact of those same factors. What we have not done is intervene on the attention structure itself: we do not ablate the identified entity heads and measure the resulting change in detection, which is the experiment that would establish those heads as the causal pathway rather than a correlate of it. The link from attention specialization to detection performance therefore remains associational, and the analysis is conducted on the enterprise dataset alone. Additional analyses including GFM signal retention, embedding trajectory, and attention entropy are provided in Appendix I.

M Protocol Notes: Token Budget and Statistical Reporting

Token budget and effective context

Because SecBFF substitutes continuous vectors for text, it is worth establishing that its gains do not come from a token-budget advantage, that is, from compressing entity mentions so that more events fit inside a fixed context window. They cannot, by construction. Replacement is performed in place and length-preserving: an entity mention occupying m tokens is replaced by m topology-token positions, at every occurrence in the sequence. A SecBFF input and the corresponding text-only input therefore contain exactly the same number of positions and the same number of visible events, and the two differ only in *what* occupies the entity positions, never in how many there are.

The comparison against the graph-aware baselines runs in the opposite direction from the confound. GraphPoolLinear, GraphPoolMLP, GraphToken, and GraphPrompter all retain the full textual entity name *and* prepend $K{=}4$ graph tokens, so they consume strictly more context than SECBFF for the same event sequence; Text-graph, which serializes entity neighborhoods into the prompt, consumes considerably more still. SECBFF thus achieves the best results in every setting while using the smallest input among all graph-aware variants and a context identical to text-only. Any residual token-budget effect would favor the baselines, so it cannot explain the gap.

STATISTICAL REPORTING

All numbers in Tables 1 to 3 are the mean $\pm$ standard deviation over three random seeds applied to the classification-adapter phase; the standard deviation is the sample standard deviation across those three runs. Every baseline is rerun under the same three seeds, the same hyperparameter search budget, and an identical compute budget, so the reported gaps are not attributable to unequal tuning effort. This matters in particular for the higher-capacity prefix baselines: GraphPrompter uses roughly $190\times$ more projector parameters than SECBFF (102M vs. 530K) yet underperforms it, and this is not a consequence of a smaller tuning budget.