

# Designing a Reliable Lateral Movement Detector Using a Graph Foundation Model

Corentin Larroche

French Cybersecurity Agency (ANSSI), Paris, France  
corentin.larroche@ssi.gouv.fr

**Abstract.** Foundation models have recently emerged as a new paradigm in machine learning (ML). These models are pre-trained on large and diverse datasets and can subsequently be applied to various downstream tasks with little or no retraining. This allows people without advanced ML expertise to build ML applications, accelerating innovation across many fields. However, the adoption of foundation models in cybersecurity is hindered by their inability to efficiently process data such as network traffic captures or binary executables. The recent introduction of graph foundation models (GFMs) could make a significant difference, as graphs are well-suited to representing these types of data. We study the usability of GFMs in cybersecurity through the lens of one specific use case, namely lateral movement detection. Using a pre-trained GFM, we build a detector that reaches state-of-the-art performance without requiring any training on domain-specific data. This case study thus provides compelling evidence of the potential of GFMs for cybersecurity.

**Keywords:** Intrusion detection · Anomaly detection · Graph neural networks.

## 1 Introduction

An immediate and lasting impact of large language models [9,42] (LLMs) and their multimodal counterparts, such as vision-language models [29], has been the democratization of machine learning (ML) application development. Consider for instance natural language processing: whereas building a text classifier used to require gathering high-quality labeled data, then designing and training a model, it can now be achieved by simply sending the right prompt to a pre-existing LLM and extracting information from its answer. By allowing domain experts without advanced ML skills to build ML applications, this new paradigm accelerates the adoption of ML in various domains.

More generally, this trend results from the emergence of **foundation models**—that is, models pre-trained on large and diverse datasets, that can be used for various tasks with little or no retraining. Beyond LLMs, foundation models have been developed for tasks ranging from time series forecasting [2] to image segmentation and classification [24,34]. However, their contribution to cybersecurity-related applications has so far been limited because these applications often deal with peculiar data. Unlike textual documents, classifying binary

executables or network traffic captures using LLMs is impractical as they differ too much from typical LLM training data. On the other hand, graphs are well-suited to representing such data, and graph neural networks (GNNs) are thus often used in cybersecurity-related ML applications [4,5].

In the last few years, pre-trained GNNs specialized on various graph-related tasks have started to emerge. These **graph foundation models** [31] (GFMs) could make it significantly easier to build ML applications for many cybersecurity-related use cases, in the same way that LLMs democratized natural language processing. In previous work [25], we performed an initial evaluation of GFMs for a specific cybersecurity problem, namely lateral movement detection. This task is well-suited to graph-based approaches as it can be framed as detecting anomalous edges in a graph [8,23]. Here, we extend this work into a detailed case study on the following question: **can we reach state-of-the-art performance in lateral movement detection using a pre-trained GFM, only through careful input graph construction and output post-processing?** By imposing this constraint, we aim to assess the ability of cybersecurity practitioners without ML expertise to build effective ML applications using GFMs. Through experiments on two benchmark datasets, we show that a GFM-based detector can indeed outperform the state-of-the-art GNN-based algorithm ARGUS [45]. This demonstrates the potential of GFMs in cybersecurity-related use cases, opening the way for further research on other cybersecurity problems involving graph-structured data.

In summary, we make the following contributions:

- We extend previous work on GFM-based lateral movement detection, improving the design of the detector with an input graph construction module that retrieves the most relevant contextual data and an output post-processing module that leverages domain knowledge on lateral movement to eliminate false positives.
- We thoroughly demonstrate the impact of these improvements on detection performance through experiments on two benchmark datasets.
- Overall, we provide evidence that GFMs can already be used in valuable cybersecurity-related applications, motivating further research on this topic.

The rest of this paper is structured as follows. We start with necessary background on lateral movement detection and graph foundation models in Section 2. Then, we describe the design of our GFM-based detector in Section 3, and report on our experiments in Section 4. Finally, we discuss the limitations and possible extensions of our work in Section 5.

## 2 Background

We first provide some necessary background and definitions. Section 2.1 defines lateral movement detection, then Section 2.2 reviews past research on this problem. Finally, we provide an introduction to graph foundation models in Section 2.3, focusing on ULTRA [15], which is the model we use in this work.

## 2.1 Problem Statement — Lateral Movement Detection

**Lateral movement** is a widespread offensive tactic that consists in propagating from one compromised host to another within the target network. It is a critical phase of advanced, multi-step intrusions as it allows the attacker to explore the network and move closer to their end goal (e.g., stealing sensitive data from a given server or deploying ransomware on all hosts). Many techniques and procedures exist for lateral movement, often leveraging legitimate tools and credentials [1]. This diversity makes detecting lateral movement challenging.

Internal network traffic and authentication logs are useful data sources in the search for lateral movements<sup>1</sup>. Such data can be efficiently represented as a graph whose nodes are the hosts of an enterprise network, with edges standing for information flows between hosts. Additional nodes representing user accounts can also be included in this graph when authentication logs are available. The graph is typically directed, and it can also be made temporal (network flows and authentication events are timestamped) and heterogeneous (there are several edge types representing, for instance, network protocols or authentication packages). Lateral movement then results in unexpected edges in this graph. From a mathematical perspective, lateral movement detection can thus be phrased as a **graph-based anomaly detection** problem.

*Key definitions and notations.* We consider a **graph sequence**  $(\mathcal{G}_t)_{t \geq 0}$ , where each graph  $\mathcal{G}_t = (\mathcal{V}_t, \mathcal{E}_t, \mathcal{R}_t)$  represents events recorded in a time window  $t$  and is defined by its node set  $\mathcal{V}_t$ , edge set  $\mathcal{E}_t$  and relation (or edge type) set  $\mathcal{R}_t$ . Edges, defined by a source node  $u$ , an edge type  $r$ , and a destination node  $v$ , are denoted  $(u, r, v) \in \mathcal{V}_t \times \mathcal{R}_t \times \mathcal{V}_t$ . Note that such heterogeneous graphs are typically called **knowledge graphs** in the literature. The goal of lateral movement detection is to build an **anomaly scoring function**  $s$  such that  $s(u, r, v; t)$  is large if the edge  $(u, r, v)$  is anomalous in the graph  $\mathcal{G}_t$ . Finally, we define the **union** of two graphs as the graph whose node, edge, and relation sets are the union of theirs.

## 2.2 Related Work — From Heuristics to Graph Neural Networks

We now review the main approaches to lateral movement detection in the literature, distinguishing three main categories: heuristics, linear and multilinear link predictors, and graph neural networks.

*Heuristics and pattern mining.* Early work on lateral movement detection uses rather simple models and heuristics. In particular, a straightforward way to define anomalous edges in a temporal graph is to count the occurrences of each edge over time. The **edges that appear the least frequently** are then deemed anomalous. Additional edge features, such as edge type or timestamp, can also be taken into account when looking for rare edges [7]. The main shortcoming of this basic heuristic is its tendency to generate many false positives, as rare edges

<sup>1</sup> Note that our threat model assumes the integrity of these data sources.

are often observed due to benign errors or legitimate behaviors. A typical way to mitigate this problem is to look for **clusters of rare edges** sharing a common source node [32,33,43,7], since attackers often use one compromised host as a stepping stone to explore the rest of the network. Rare connections between hosts can also be correlated with other possible signs of lateral movement, such as the creation of a new service on the target host [30]. Alternatively, simple statistical models can be used to characterize normal rare edges, thus weeding out false positives [18], or to detect anomalous paths in the host communication graph [6]. Finally, a closely related approach consists in mining frequent patterns among edges instead of frequent edges [39]. These patterns then also cover rarely observed edges that share some distinctive traits with frequent edges, ensuring that such rare but legitimate edges are not mistakenly flagged as malicious.

*Linear and multilinear models.* Detecting anomalous edges in a graph is in fact complementary to the well-studied problem of **link prediction**, i.e., predicting which new edges are the most likely to appear in a given graph. Since anomalous edges are those least likely to appear, a good link predictor can also be used as an anomaly detector. Several models that were initially designed for link prediction, typically for recommender systems, were thus subsequently used for lateral movement detection. These models include matrix factorization [37], tensor factorization [12,13], factorization machines [41] and latent space models [26], which are all multilinear models. More specifically, these models compute a latent vector for each node during training, then define the probability of an edge as a function of the dot product between its endpoints' latent vectors. A closely related approach relies on graph embedding algorithms such as node2vec [16], which also compute a latent vector (also called embedding) for each node. Linear models can then take these vectors as input in order to predict normal edges and/or detect anomalous ones [44,50,8,35].

*Graph neural networks.* Finally, more recent and sophisticated link prediction algorithms go beyond linear models and use multilayer graph neural networks [10]. As a consequence, research on lateral movement detection has also adopted GNNs. Early contributions treated internal network traffic or authentication logs as a static graph, training standard GNNs on all past data to predict future edges [28,40]. Subsequent work introduced a **temporal perspective** by representing connections between internal hosts as a sequence of graphs. This sequence can then be modeled by intertwining a GNN and a recurrent neural network (RNN), giving the model the ability to detect temporal anomalies in addition to structural ones [23,22,45]. The current state-of-the-art lateral movement detection algorithm, ARGUS [45], relies on this RNN-GNN combination.

### 2.3 Graph Foundation Models for Link Prediction

Following the emergence of powerful foundation models for modalities such as text and images, the graph machine learning community has started exploring the possibility of building such foundation models for graph-related tasks. As a

result, such models have been proposed for various tasks, including node classification [19,49], node anomaly detection [36], and link prediction [11,38]. In particular, **link prediction in knowledge graphs** has been addressed by several GFMs [15,46]. This is especially interesting for lateral movement detection, which can be framed as detecting anomalous edges in a knowledge graph.

This work focuses on ULTRA [15], a GFM designed for link prediction in knowledge graphs. ULTRA builds upon the Neural Bellman-Ford Network [51] (NBFNet) architecture to predict possible edges in any knowledge graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{R})$  through the following steps. First, reciprocal edges are added to the graph  $\mathcal{G}$ : for each relation  $r \in \mathcal{R}$ , a reciprocal relation  $r^{-1}$  is defined and a new edge  $(v, r^{-1}, u)$  is created for each existing edge  $(u, r, v) \in \mathcal{E}$ . Then, a graph of relations  $\mathcal{G}_R = (\mathcal{R}, \mathcal{E}_R, \mathcal{R}_R)$  is built. Its nodes are the relations of the original knowledge graph  $\mathcal{G}$ , and an edge from relation  $r$  to relation  $r'$  indicates that there exist two edges in  $\mathcal{G}$ , with respective types  $r$  and  $r'$ , that have at least one node in common. This common node can be either the source or destination of each of the two edges, thus there are four possible types of interactions between relations  $r$  and  $r'$  (i.e.,  $|\mathcal{R}_R| = 4$ ). A first NBFNet is trained to compute embeddings of the relations in  $\mathcal{R}$  given the graph of relations  $\mathcal{G}_R$ . Then, given the original graph  $\mathcal{G}$  and these relation embeddings, a second NBFNet predicts the probability of any edge  $(u, r, v) \in \mathcal{V} \times \mathcal{R} \times \mathcal{V}$ . The authors of ULTRA **pre-trained** three such models on a set of knowledge graphs representing knowledge bases from several domains. They showed that these models achieved competitive link prediction performance on knowledge graphs not included in their training set, which makes them foundation models for link prediction in knowledge graphs.

From a high-level perspective, a pre-trained ULTRA model is a function that takes as input a knowledge graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{R})$  (which we call the **context graph**) and a triple  $(u, r, v) \in \mathcal{V} \times \mathcal{R} \times \mathcal{V}$ , and returns a score  $g(u, r, v; \mathcal{G})$  that is large if the edge  $(u, r, v)$  is likely to appear in the graph  $\mathcal{G}$ . In previous work [25], we showed that the pre-trained models introduced in the ULTRA paper [15] could be used for lateral movement detection without retraining, achieving competitive detection performance with respect to standard multilinear models. Here, we extend this work by designing a better detector using the ULTRA50G model. This GFM has 177K parameters and is openly available on GitHub<sup>2</sup> and HuggingFace<sup>3</sup>. Our hypothesis is that significant performance gains can be unlocked using the same pre-trained model, only through changes in context graph construction and output post-processing—an approach inspired by LLM-based applications.

### 3 Design of Our Lateral Movement Detector

We improve upon our initial algorithm, ULTRALMD [25], by drawing inspiration from the state-of-the-art GNN-based detector ARGUS [45]. Through this approach, we investigate the possibility of replicating the performance of a specially designed GNN by cleverly querying a pre-trained GFM. We provide an

<sup>2</sup> <https://github.com/DeepGraphLearning/ULTRA>

<sup>3</sup> [https://huggingface.co/mgalkin/ultra\\_50g](https://huggingface.co/mgalkin/ultra_50g)

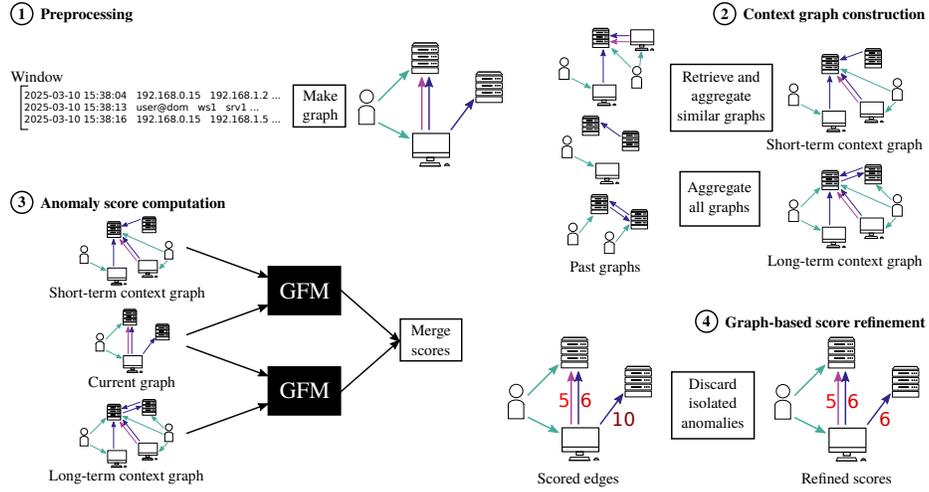


Fig. 1. High-level workflow of ULTRALMD++.

overview of our new detector, called ULTRALMD++, in Section 3.1, then describe its main components in further detail: context graph construction (Section 3.2), anomaly scoring (Section 3.3), and output post-processing (Section 3.4).

### 3.1 Overview

Given a sequence of authentication events and/or network flows, our lateral movement detector applies the following steps, illustrated in Figure 1. First, the events are split into **fixed-length windows**. Then, for each time window  $t$ , we build a graph  $\mathcal{G}_t$  representing the events within this window. In order to compute anomaly scores for the edges in  $\mathcal{G}_t$ , we first need to build a **context graph**  $\mathcal{H}_t$  using **past events** (Section 3.2). This context graph can then be passed as input to the GFM, which predicts how likely the edges in  $\mathcal{G}_t$  are given the context graph  $\mathcal{H}_t$ . **Anomaly scores** are derived from these predictions (Section 3.3). Finally, these anomaly scores are **refined** by leveraging the **structure of the graph**  $\mathcal{G}_t$ , building upon the idea that lateral movement edges tend to be clustered into connected regions of the network (Section 3.4).

### 3.2 Context Graph Construction

At each time step  $t$ , we observe a new graph  $\mathcal{G}_t = (\mathcal{V}_t, \mathcal{E}_t, \mathcal{R}_t)$ . Our goal is to compute an anomaly score for each edge in  $\mathcal{E}_t$  by comparing the graph  $\mathcal{G}_t$  with past graphs  $\{\mathcal{G}_{t'}; t' < t\}$ . ARGUS [45] adopts a temporal modeling approach to incorporate the fact that **a given edge might be normal at some time steps and anomalous at others** (consider for instance a remote authentication to a server at 2p.m. on a Monday versus at 11p.m. on a Saturday). To that end, it

uses a combined RNN-GNN model to encode the sequence of past graphs and make time-dependent predictions for the probability of each edge in  $\mathcal{E}_t$ . Since we use a GFM to score edges, we cannot modify its architecture to include an RNN. We thus adopt a different approach, which focuses on the context graph used by the GFM to predict new edges.

As explained in Section 2.3, we consider a GFM that takes as input a context graph and a possible edge, and predicts the probability of that edge appearing in the context graph. **Time-dependent information** can thus be passed to the GFM through the **construction of this context graph**. Specifically, given the current graph  $\mathcal{G}_t$  and the sequence of past graphs  $\{\mathcal{G}_{t'}; t' < t\}$ , we build two context graphs  $\mathcal{H}_t^L$  and  $\mathcal{H}_t^S$  encoding long-term and short-term context, respectively. The long-term context graph is simply the union of all past graphs,  $\mathcal{H}_t^L = \bigcup_{t' < t} \mathcal{G}_{t'}$ . As for the short-term context graph  $\mathcal{H}_t^S$ , we build it using a method analogous to retrieval-augmented generation (RAG [27]) for LLMs. Given a **similarity function**  $c$  that compares two graphs, we compute the similarity between the current graph  $\mathcal{G}_t$  and each past graph. Denoting  $\mathcal{C}$  the set of indices corresponding to the  $K$  past graphs most similar to  $\mathcal{G}_t$  (for some fixed integer  $K$ ), the short-term context graph is then defined as  $\mathcal{H}_t^S = \bigcup_{t' \in \mathcal{C}} \mathcal{G}_{t'}$ .

The similarity function  $c$  we use in our experiments is simple and computationally inexpensive. It is defined as the sum of two components,  $c = c_V + c_E$ , where  $c_V$  is the Jaccard index of the respective node sets of the two graphs,

$$c_V(\mathcal{G}_t, \mathcal{G}_{t'}) = \frac{|\mathcal{V}_t \cap \mathcal{V}_{t'}|}{|\mathcal{V}_t \cup \mathcal{V}_{t'}|},$$

and  $c_E$  measures the similarity of the two graphs in terms of edge types. Let  $\theta_r(\mathcal{G}_t) = |\{(u, r', v) \in \mathcal{E}_t : r' = r\}|$  be the number of edges of type  $r$  in the graph  $\mathcal{G}_t$ . Then the relation similarity function  $c_E$  is defined as

$$c_E(\mathcal{G}_t, \mathcal{G}_{t'}) = \frac{\tilde{\theta}(\mathcal{G}_t) \cdot \tilde{\theta}(\mathcal{G}_{t'})}{\|\tilde{\theta}(\mathcal{G}_t)\| \|\tilde{\theta}(\mathcal{G}_{t'})\|}, \quad \tilde{\theta}(\mathcal{G}) = \left( \frac{\theta_r(\mathcal{G}) - \bar{\theta}_r}{\sigma_r} \right)_{r \in \mathcal{R}_t \cup \mathcal{R}_{t'}},$$

where  $\bar{\theta}_r$  and  $\sigma_r$  denote the mean and standard deviation of  $\theta_r$  over the set of graphs  $\{\mathcal{G}_{t'}; t' \leq t\}$ , respectively. In other words,  $c_E$  is the cosine similarity between the standardized vectors of relation counts of the two graphs.

### 3.3 Anomaly Score Computation

Given the current graph  $\mathcal{G}_t$  and the short and long-term context graphs  $\mathcal{H}_t^S$  and  $\mathcal{H}_t^L$ , we can now use the GFM to compute anomaly scores for the edges of  $\mathcal{G}_t$ . As stated in Section 2.3, the GFM is a function that computes a score  $g(u, r, v; \mathcal{H})$  given a context graph  $\mathcal{H} = (\mathcal{V}_{\mathcal{H}}, \mathcal{E}_{\mathcal{H}}, \mathcal{R}_{\mathcal{H}})$  and an edge  $(u, r, v) \in \mathcal{V}_{\mathcal{H}} \times \mathcal{R}_{\mathcal{H}} \times \mathcal{V}_{\mathcal{H}}$ , with higher scores meaning more likely edges. We turn these scores into anomaly scores as follows. First, we define the **predicted probability** of the destination node  $v$  given the source  $u$ , the edge type  $r$ , and the context graph  $\mathcal{H}$  as

$$p(v \mid u, r, \mathcal{H}) = \frac{\exp(g(u, r, v; \mathcal{H}))}{\sum_{v' \in \mathcal{V}_{\mathcal{H}}} \exp(g(u, r, v'; \mathcal{H}))}.$$

The probability of the source node  $u$  given the destination  $v$ , the edge type  $r$ , and the context graph  $\mathcal{H}$  can be defined in a similar way using the reciprocal relation  $r^{-1}$ , defined in Section 2.3. The **anomaly score** of the edge  $(u, r, v)$  given the context graph  $\mathcal{H}$  is then defined as

$$s(u, r, v; \mathcal{H}) = -\log(\min\{p(v | u, r, \mathcal{H}), p(u | v, r^{-1}, \mathcal{H})\}).$$

Two specific cases are handled differently. First, if  $(u, r, v) \in \mathcal{E}_{\mathcal{H}}$ , then we set  $s(u, r, v; \mathcal{H}) = 0$ . In other words, **known edges are considered normal**. Note that while the long-term context graph contains all previously observed edges, the short-term context graph only contains edges from the past graphs most similar to the current graph. As a consequence, an edge that was previously observed in a different context can still be declared anomalous in the present context by our detector. The other specific case arises when scoring an edge whose **type does not exist** in the context graph, that is,  $r \notin \mathcal{R}_{\mathcal{H}}$ . Since the GFM is unable to compute scores for such unknown edge types, we then define the anomaly score as the average over the set of known edge types  $\mathcal{R}_{\mathcal{H}}$ . With these definitions, the anomaly score for an edge  $(u, r, v)$  given the short and long-term context graphs  $\mathcal{H}_t^S$  and  $\mathcal{H}_t^L$  is

$$s(u, r, v; t) = \frac{1}{2} (s(u, r, v; \mathcal{H}_t^S) + s(u, r, v; \mathcal{H}_t^L)). \quad (1)$$

### 3.4 Graph-Based Score Refinement

In addition to individually scoring each edge in the current graph, looking for **connected clusters of anomalous edges** is a widespread approach in the lateral movement detection literature [32,8,26,7]. The motivation for this strategy is that attackers typically perform several lateral movements originating from the same compromised host, whereas rare but legitimate activity is randomly scattered across the internal network. ARGUS leverages this domain knowledge through the design of its decoder (i.e., the last GNN layer that predicts likely edges). Once again, since we build upon a pre-trained model, we cannot change its architecture and must therefore look for a different method.

We draw inspiration from Bowman et al. [8], who proposed **discarding all anomalous edges that do not share a node with at least one other anomalous edge**. This idea can be translated into the following edge score refinement method, formally described in Algorithm 1 of Appendix A. For each edge  $e$  in the current graph  $\mathcal{G}_t$ , we retrieve the anomaly scores (as defined in Equation 1) of all edges in  $\mathcal{G}_t$  that share at least one node with  $e$ . Then, there are two possibilities: either there is at least one edge in this set with a higher anomaly score than  $e$ , thus for any detection threshold low enough to declare  $e$  anomalous,  $e$  will be adjacent to another anomalous edge; or  $e$  is the highest-scored edge within its neighborhood. In that second case, we set the score of  $e$  to the maximum of the scores of adjacent edges, so that  $e$  is only declared anomalous when there is at least one other anomalous edge within its neighborhood. In practice, this score refinement method can be implemented efficiently using a message passing algorithm.

## 4 Experiments

We now empirically study the behavior and performance of ULTRALMD++, specifically investigating the following questions. First, since our goal is to build a lateral movement detector that is at least as reliable as the state-of-the-art algorithm ARGUS [45], we compare their respective **detection performance** on two public benchmark datasets. Secondly, since ULTRALMD++ differs from the original ULTRALMD detector in two aspects (namely, context graph retrieval and score refinement), we are interested in evaluating the **exact contribution of each of these two improvements**. Third, an expected benefit of using a GFM-based lateral movement detector is that it can handle concept drift without retraining. Thus we investigate the temporal dynamics of the distribution of anomaly scores to assess the **robustness of ULTRALMD++ to concept drift**. Finally, the **computational cost** of ULTRALMD++ is an important aspect of its real-world usability. We thus study the run time of each of its components (i.e., context graph construction, score computation, and score refinement).

We implemented ULTRALMD++ in Python and make the code openly available<sup>4</sup>. As for ARGUS, we used the implementation provided by the authors<sup>5</sup>. We run our experiments on a 2.2GHz, 40-core CPU with 384GB of RAM, and an Nvidia GeForce RTX 2080 Ti GPU with 11GB of memory. The datasets and evaluation metrics used in our experiments are described in Section 4.1. We then present our results on the two datasets in Sections 4.2 and 4.3, respectively.

### 4.1 Datasets and Metrics

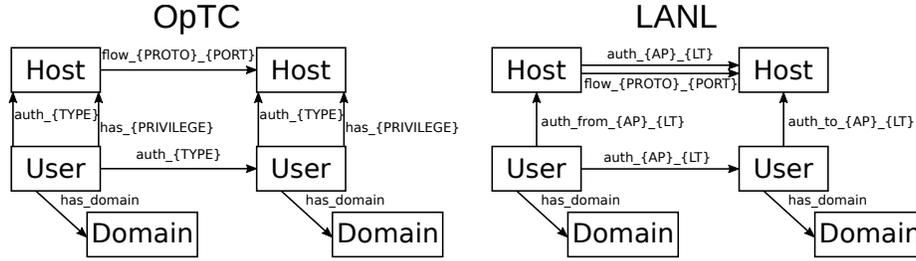
We compare ULTRALMD++ and ARGUS on two benchmark datasets: the "Operationally Transparent Cyber" (**OpTC**) dataset released by DARPA in 2020, and the "Comprehensive, Multi-Source Cyber-Security Events" (**LANL**) dataset released in 2015 by the Los Alamos National Laboratory [21,20]. OpTC consists of host and network logs generated over nine days in a simulated enterprise network. Three distinct attacks were carried out by pentesters against this simulated network, two of which comprise a lateral movement phase. The first six days are used for training ARGUS, and both ARGUS and ULTRALMD++ are evaluated on the last three days, during which the attacks take place. As for the LANL dataset, it contains anonymized host logs and network flows collected over 58 days within LANL's enterprise network. A red team exercise took place during this time frame, and lateral movement events are labeled. Since the first of these events occurs during the 42<sup>nd</sup> hour, we use the first 41 hours for training ARGUS and the remaining data for evaluation. Note that while the authors of ARGUS used the same datasets in their experiments, we perform different preprocessing steps here, which leads to discrepancies with respect to the results reported in the ARGUS paper [45]. See Appendix B for more details.

<sup>4</sup> <https://github.com/cl-anssi/UltraLMD>

<sup>5</sup> <https://github.com/C0ldstudy/Argus>

**Table 1.** Datasets used in our experiments. LM stands for lateral movement edges; Min./Med./Max. is the minimum/median/maximum number of edges per time step.

Dataset	Hosts	Users	Time steps	Edges			Types	LM
				Min.	Med.	Max.		
OpTC	898	1,505	2,113	0	5,795	17,782	39	626
LANL	16,377	25,701	1,392	72,717	151,343.5	286,350	106	500

**Fig. 2.** Knowledge graph representation of the two datasets for ULTRALMD++. AP and LT stand for authentication package and logon type, respectively. Nodes and edges representing node types are not displayed for the sake of readability.

*Graph construction.* Both ARGUS and ULTRALMD++ require representing the datasets as sequences of graphs. First of all, we chunk both datasets into fixed-length time windows. We use the same window lengths as the ARGUS paper, namely one hour for the LANL dataset and six minutes for the OpTC dataset. For ARGUS, we then use the same graph construction method as the original paper. As for ULTRALMD++, the graphs are built as follows. For each remote authentication event, we create an edge from the source host to the destination host. The type of the edge is the pair (authentication package, logon type). We also create host-host edges for network flow events, whose type is the pair (transport protocol, destination port). To limit the number of edge types, only the 20 flow types most frequently seen during the training period are kept, and all other types are replaced with a single "Other" type. We add user-host edges for each authentication event, which indicate that the user has logged on from or to the host. Finally, we define five node types: user account, computer account, built-in account (e.g., SYSTEM or LOCAL SERVICE), computer, and domain. These types are represented as nodes and linked to other nodes with edges of a special type. See Figure 2 for a summary of these graph representations, and Table 1 for a summary of the datasets. Note that for consistency with ARGUS, only host-host edges representing network flows in the OpTC dataset and remote authentications in the LANL dataset are given anomaly scores.

*Evaluation metrics.* We compare the detection performance of ARGUS and ULTRALMD++ using the following metrics. First, the area under the ROC curve (**AUC**) is a standard metric for binary classification, which captures the trade-

**Table 2.** Results of our experiments on OpTC.

Algorithm	AUC	AP	Rec@3	Rec@5	Rec@10
ARGUS [45]	.9300	.0240	.0064	.0113	.0273
ULTRALMD [25]	.9826	.1003	.0208	.0319	.0559
ULTRALMD + retrieval	<b>.9909</b>	.1506	<b>.0256</b>	.0367	.0607
ULTRALMD + refinement	.9826	.1003	.0192	.0319	.0559
ULTRALMD++	<b>.9909</b>	<b>.1510</b>	<b>.0256</b>	<b>.0383</b>	<b>.0623</b>

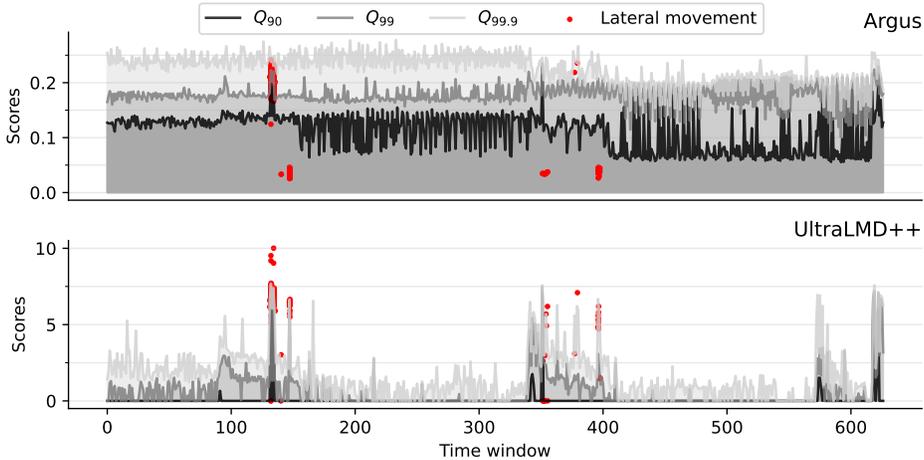
off between the true positive rate (TPR, also called recall) and the false positive rate (FPR) at various detection thresholds. Since the AUC tends to be overly optimistic in highly imbalanced settings (base rate fallacy [3]), we also compute the average precision (**AP**), which evaluates the ratio between the number of true and false positives at different true positive rates. This metric puts more emphasis on low-FPR settings, which corresponds to the operational constraints of intrusion detection. Finally, to get a more realistic picture of each detector’s reliability in a real-world deployment, we compute the recall for various detection budgets  $B$  (**Rec@ $B$** ). In other words,  $\text{Rec}@B$  is the proportion of lateral movement edges that are detected when the top  $B$  most anomalous edges in each time window are investigated.

## 4.2 Results on OpTC

We run both ARGUS and ULTRALMD++ on the OpTC dataset, using the same hyperparameters as the authors for ARGUS and setting the number  $K$  of past graphs to include in the short-term context of ULTRALMD++ to 100. The results of these experiments are shown in Table 2. We now discuss them in further detail.

First of all, **ULTRALMD++ consistently outperforms ARGUS across all metrics**, highlighting the ability of GFM-based lateral movement detectors to compete with GNNs that are specifically trained for this task. We emphasize that ULTRA50G, which underpins ULTRALMD++, was not trained on any cybersecurity-related data, whereas ARGUS was trained on the first 6 days of the OpTC dataset. This makes the performance of ULTRALMD++ especially compelling. Note that the recall at reasonable detection budgets remains rather low for both detectors. However, real-world intrusions often comprise several lateral movements, and detecting some of them can suffice to catch the whole attack. This principle stands for the OpTC dataset, where even detecting 5% of the hundreds of lateral movement edges is enough to make a detector useful.

As for the ablation study, while ULTRALMD++ clearly outperforms ULTRALMD, **most of the improvement comes from the retrieval component**. Only adding score refinement leads to almost identical, and in fact slightly worse performance. This limited impact of score refinement might result from the short length of the time window used for the OpTC dataset: with each graph representing a six-minute window, lateral movement edges are more likely to be scattered

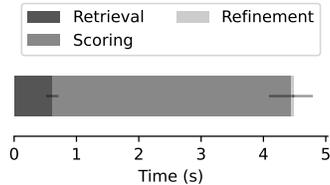


**Fig. 3.** Evolution of the 90<sup>th</sup>, 99<sup>th</sup>, and 99.9<sup>th</sup> percentiles of the distribution of anomaly scores over time on OpTC, for ARGUS and ULTRALMD++.

across several graphs. As a consequence, graph-based score refinement can actually reduce their anomaly scores as much as those of benign edges. While this does not refute the general usefulness of score refinement, it does point to unfavorable settings that can make it less effective. Still, the performance gain brought by context graph retrieval sustains our main hypothesis: careful input design can yield superior detection performance out of the same GFM.

Regarding score dynamics, Figure 3 shows the evolution of the distribution of anomaly scores (90<sup>th</sup>, 99<sup>th</sup>, and 99.9<sup>th</sup> percentiles) over time, as well as the scores of lateral movement edges, for ARGUS and ULTRALMD++. Due to the limited duration (three days) of the OpTC test set, no conclusion can be drawn regarding each detector’s propensity to concept drift from this dataset. However, we observe that ARGUS performs better on the first attack (around window 130) than on the second (windows 350-400), while ULTRALMD++ predicts relatively high anomaly scores for lateral movement edges in both attacks. This further demonstrates the superior reliability of ULTRALMD++ on the OpTC dataset.

Finally, Figure 4 shows the average run time of ULTRALMD++ for a single time window on OpTC. With only a few seconds of computation for a six-minute window, **ULTRALMD++ is efficient enough to run in real time**. Note that the anomaly scoring step makes up for most of the computational cost, which could thus be significantly reduced with more powerful GPUs.



**Fig. 4.** Run time per window (mean and standard deviation) of ULTRALMD++ on OpTC.

**Table 3.** Results of our experiments on LANL, with and without duplicate lateral movement edges.

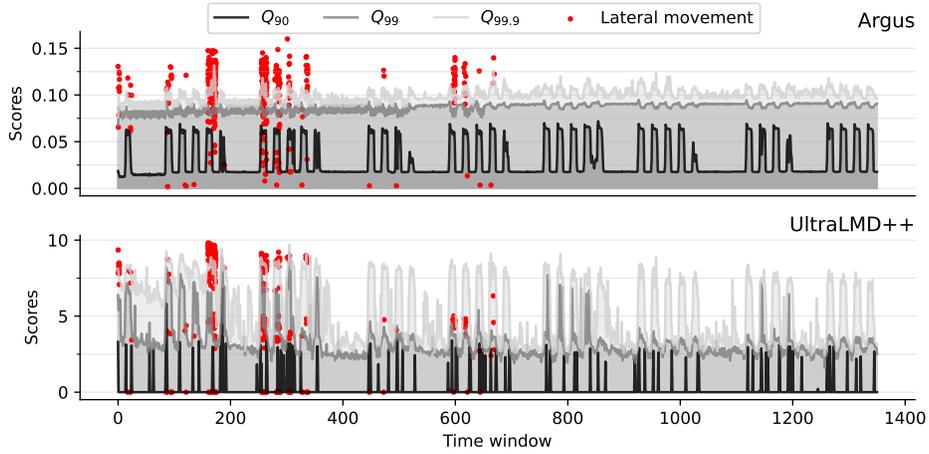
Duplicates	Algorithm	AUC	AP	Rec@3	Rec@5	Rec@10
yes	ARGUS [45]	<b>.9749</b>	<b>.0065</b>	.0100	.0260	.0900
	ULTRALMD [25]	.7645	.0020	.0260	.0400	.0720
	ULTRALMD + retrieval	.8969	.0040	.0300	.0480	.0860
	ULTRALMD + refinement	.7606	.0026	<b>.0360</b>	.0460	.0820
	ULTRALMD++	.8938	.0056	.0340	<b>.0560</b>	<b>.1040</b>
no	ARGUS [45]	.9821	.0056	.0166	.0364	.1126
	ULTRALMD [25]	.9394	.0034	.0430	.0662	.1192
	ULTRALMD + retrieval	<b>.9920</b>	.0063	.0497	.0795	.1424
	ULTRALMD + refinement	.9328	.0042	.0596	.0762	.1358
	ULTRALMD++	.9868	<b>.0088</b>	<b>.0629</b>	<b>.0927</b>	<b>.1788</b>

### 4.3 Results on LANL

Similarly to our experiments on OpTC, we also run ARGUS with the same hyperparameters as the authors on the LANL dataset. As for ULTRALMD++, we set the number  $K$  of past graphs to include in the short-term context to 10. Note that many lateral movement edges are repeated several times in the LANL dataset (i.e., the red team repeated the same lateral movement at different timestamps). Since detecting a single occurrence of each lateral movement edge can be considered sufficient, we also compute the performance metrics after removing duplicates. Specifically, for each evaluated detector, we only keep the highest-scoring instance of each unique lateral movement edge. The results of our experiments with and without this deduplication step are shown in Table 3.

Starting with the results **without deduplication**, ULTRALMD++ **performs best in terms of recall at fixed detection budgets**, while ARGUS performs best in terms of AUC and AP. In contrast, **ULTRALMD++ outperforms ARGUS across all metrics when removing duplicate lateral movement edges**. The reason for this discrepancy is that we designed our detector to treat edges that appear in both the short and long-term context graphs as normal. As a consequence, even though lateral movement edges generally get a high anomaly score on their first occurrence, they are then included in the context graphs when they reoccur. However, since detecting a lateral movement edge on its first occurrence is arguably good enough, this is not a major flaw. Besides, even without deduplication, ULTRALMD++ outperforms ARGUS with respect to the recall at fixed detection budgets, which more closely mirrors the constraints of real-world deployments. ULTRALMD++ can thus be considered more effective than ARGUS. Again, we emphasize that this level of performance is reached using a GFM that was not trained for lateral movement detection.

The **respective contributions** of context graph retrieval and anomaly score refinement to detection performance are **more evenly distributed** than in the OpTC experiment. Adding either one of these components increases almost all

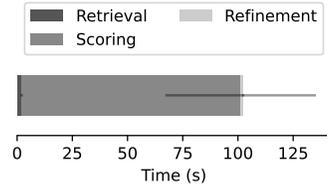


**Fig. 5.** Evolution of the 90<sup>th</sup>, 99<sup>th</sup>, and 99.9<sup>th</sup> percentiles of the distribution of anomaly scores over time on LANL, for ARGUS and ULTRALMD++.

metrics, and adding both is even better, allowing ULTRALMD++ to consistently outperform ARGUS (whereas ULTRALMD does worse than ARGUS by several metrics). Overall, this demonstrates the usefulness of the two improvements we bring to the original ULTRALMD detector, further confirming that adequately querying a pre-trained GFM has a significantly positive impact.

The evolution of the distribution of anomaly scores over time and the scores of lateral movement edges for ARGUS and ULTRALMD++ are displayed in Figure 5. **The distribution remains rather stable for ULTRALMD++**, except for a downward trend in the first 400 windows. This trend is expected: as more and more past graphs become available, the context graphs used for anomaly scoring become denser and more similar to the current graph, thus the new edges appear less and less anomalous. In contrast, **evidence of drift can be observed for ARGUS**. In particular, the 99<sup>th</sup> and 99.9<sup>th</sup> percentiles of the distribution exhibit a slight upward trend, suggesting that the model becomes less and less well-adjusted to the data distribution as the new graphs start becoming less similar to those in the training set. While this problem could typically be addressed by retraining the model on more recent data, the need for frequent retraining is impractical in real-world settings and can hinder the deployment of ML-based detectors. The ability of ULTRALMD++ to straightforwardly adapt to distribution shifts is therefore a significant upside.

Finally, Figure 6 shows the average run time of ULTRALMD++ for a single time window on LANL. Similarly to OpTC, **the run time is low enough**



**Fig. 6.** Run time per window (mean and standard deviation) of ULTRALMD++ on LANL.

**for ULTRALMD++ to be run in real time** (less than two minutes per one-hour window). This time is also mostly spent on anomaly scoring. Overall, the ability of ULTRALMD++ to run in real time on a 16K-host enterprise network using rather affordable hardware demonstrates the efficiency of GFMs, which are typically orders of magnitude smaller than LLMs in terms of parameter count. This efficiency is another upside of GFMs for cybersecurity-related applications, which often deal with sensitive data and must therefore be run on premise.

## 5 Discussion

We now discuss the limitations of our work as well as its possible extensions and implications for future research.

*Future work on GFM-based lateral movement detection.* First of all, ULTRALMD++ could still be improved in several directions. In particular, whether **fine-tuning a pre-trained GFM** could improve detection performance remains an open question. In that regard, a noteworthy aspect of ARGUS is that it was trained with a specific loss function, which was shown to yield better results [45]. Fine-tuning GFMs with carefully designed loss functions might thus be a lead worth investigating. Future work could also focus on the **retrieval component** of our detector: first, the similarity function could be improved so that more relevant context graphs are retrieved. Secondly, a current limitation of ULTRALMD++ is that it needs to observe an entire graph in order to compute its similarity to past graphs and score its edges. Malicious edges can thus only be detected once the time window they fall into is over, which can be problematic in time-critical situations. A possible solution could be to train a model to predict which graphs are the most relevant at a given time, so that these graphs can be retrieved in advance. This approach is similar to the one used by Gutflaish et al. [17] to dynamically rescale anomaly scores in the context of user behavior analysis. Finally, **experimenting with other GFMs**, such as the more recently introduced TRIX [46], could also increase detection performance.

*Further use of GFMs in cybersecurity.* Beyond lateral movement detection, there is significant potential for useful cybersecurity-related applications of GFMs. A straightforward way to identify such applications is to consider past research on using GNNs in this field. This includes host and network intrusion detection [4], malware detection [5], and report analysis for cyber threat intelligence [14]. The benefits brought by GFMs in such cases could be the following. First, by making it possible to develop new applications without training any model, they could **accelerate innovation**. In particular, lowering the required ML expertise would allow more domain experts to build graph ML-based applications tailored to their own needs. Secondly, in the specific case of intrusion detection, GNN-based methods typically require training one model for each monitored network. This model must also be regularly retrained to avoid concept drift. In contrast, our experiments show that a single GFM can deliver high detection performance in

several different enterprise networks without any retraining. GFMs could thus make GNN-based intrusion detection **more practical** in real-world settings. Finally, applications built upon GFMs would also directly **benefit from future research** on these models: since no retraining is needed, substituting a new GFM into an existing application to study its performance is straightforward.

*GFMs and adversarial machine learning.* Besides the aforementioned opportunities, the adoption of GFMs in cybersecurity would also come with potential risks. Specifically, threats related to adversarial machine learning can be exacerbated in foundation models. For instance, creating **adversarial examples** [52] against an openly accessible GFM is easier than against a private, specifically tailored GNN. Note, however, that crafting an adversarial example against GFM-based detectors such as ULTRALMD++ would require additional knowledge besides the model itself—namely, since the anomaly scores depend on the context graph, the adversary would also need access to the data used to build this context graph. Risks related to adversarial examples thus depend on the application. **Data poisoning** [47] is another potential threat that becomes more serious when relying on pre-trained models: ensuring that GFMs used in cybersecurity-related applications are trustworthy and were not trained on poisoned data would be critical. Finally, if a GFM was trained specifically for cybersecurity applications and made openly accessible, the confidentiality of its training data against **model inversion attacks** [48] would be a legitimate concern. While none of these potential threats definitely precludes the use of GFMs in the cybersecurity domain, studying them remains instrumental to the adoption of GFM-based applications.

## 6 Conclusion

The emergence of graph foundation models could be a remarkable opportunity for researchers and practitioners in cybersecurity. By removing the need to train task-specific models, GFMs shift the focus onto other aspects of ML application development, such as input construction and output post-processing. This focus shift could both make it easier for domain experts to build applications relying on graph machine learning, and make these applications more practical in real-world settings. Through a detailed case study centered around lateral movement detection, we provide evidence of this potential by demonstrating that a GFM-based detector can outperform a more traditional approach that requires training a GNN from scratch. These initial results call for further research on the use of GFMs for cybersecurity. In particular, the numerous applications of GNNs in this domain are straightforward starting points for follow-up research. Investigating the effectiveness of GFM-based systems for these various use cases will in turn enable a more thorough assessment of the impact of GFMs on the field at large.

## References

1. Lateral movement, tactic TA0008. <https://attack.mitre.org/tactics/TA0008/>, accessed: 2025-03-31

2. Ansari, A.F., Stella, L., Turkmen, A.C., Zhang, X., Mercado, P., Shen, H., Shchur, O., Rangapuram, S.S., Pineda-Arango, S., Kapoor, S., Zschiegner, J., Maddix, D.C., Mahoney, M.W., Torkkola, K., Wilson, A.G., Bohlke-Schneider, M., Wang, Y.: Chronos: Learning the language of time series. arXiv preprint arXiv:2403.07815 (2024)
3. Arp, D., Qiring, E., Pendlebury, F., Warnecke, A., Pierazzi, F., Wressnegger, C., Cavallaro, L., Rieck, K.: Dos and don'ts of machine learning in computer security. In: USENIX Security (2022)
4. Bilot, T., El Madhoun, N., Al Agha, K., Zouaoui, A.: Graph neural networks for intrusion detection: A survey. IEEE Access **11**, 49114–49139 (2023)
5. Bilot, T., El Madhoun, N., Al Agha, K., Zouaoui, A.: A survey on malware detection with graph representation learning. ACM Comput. Surv. **56**(11), 1–36 (2024)
6. Bohara, A., Noureddine, M.A., Fawaz, A., Sanders, W.H.: An unsupervised multi-detector approach for identifying malicious lateral movement. In: SRDS (2017)
7. Bowman, B., King, I.J., Melamed, R., Huang, H.H.: NetHawk: Hunting advanced persistent threats via structural and temporal graph anomalies. In: CNS (2024)
8. Bowman, B., Laprade, C., Ji, Y., Huang, H.H.: Detecting lateral movement in enterprise computer networks with unsupervised graph AI. In: RAID (2020)
9. Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., Amodei, D.: Language models are few-shot learners. In: NeurIPS (2020)
10. Corso, G., Stark, H., Jegelka, S., Jaakkola, T., Barzilay, R.: Graph neural networks. Nat. Rev. Methods Primers **4**(1), 17 (2024)
11. Dong, K., Mao, H., Guo, Z., Chawla, N.V.: Universal link predictor by in-context learning on graphs. arXiv preprint arXiv:2402.07738 (2024)
12. Eren, M.E., Moore, J.S., Alexandro, B.S.: Multi-dimensional anomalous entity detection via Poisson tensor factorization. In: ISI (2020)
13. Eren, M.E., Moore, J.S., Skau, E., Moore, E., Bhattarai, M., Chennupati, G., Alexandrov, B.S.: General-purpose unsupervised cyber anomaly detection via non-negative tensor factorization. Digit. Threat. **4**(1), 1–28 (2023)
14. Fang, Y., Zhang, Y., Huang, C.: CyberEyes: cybersecurity entity recognition model based on graph convolutional network. Comput. J. **64**(8), 1215–1225 (2021)
15. Galkin, M., Yuan, X., Mostafa, H., Tang, J., Zhu, Z.: Towards foundation models for knowledge graph reasoning. In: ICLR (2024)
16. Grover, A., Leskovec, J.: node2vec: Scalable feature learning for networks. In: KDD (2016)
17. Gutflaish, E., Kontorovich, A., Sabato, S., Biller, O., Sofer, O.: Temporal anomaly detection: calibrating the surprise. In: AAAI (2019)
18. Ho, G., Dhiman, M., Akhawe, D., Paxson, V., Savage, S., Voelker, G.M., Wagner, D.: Hopper: Modeling and detecting lateral movement. In: USENIX Security (2021)
19. Huang, Q., Ren, H., Chen, P., Kržmanc, G., Zeng, D., Liang, P.S., Leskovec, J.: Prodigy: Enabling in-context learning over graphs. In: NeurIPS (2023)
20. Kent, A.D.: Comprehensive, multi-source cyber-security events. Los Alamos National Laboratory (2015). <https://doi.org/10.17021/1179829>
21. Kent, A.D., Liebrock, L.M., Neil, J.C.: Authentication graphs: Analyzing user behavior within an enterprise network. Comput. Secur. **48**, 150–166 (2015)

22. Khoury, J., Klisura, D., Zanddizari, H., Parra, G.D.L.T., Najafirad, P., Bou-Harb, E.: Jbeil: Temporal graph-based inductive learning to infer lateral movement in evolving enterprise networks. In: SP (2024)
23. King, I.J., Huang, H.H.: EULER: Detecting network lateral movement via scalable temporal link prediction. In: NDSS (2022)
24. Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A.C., Lo, W., Dollár, P., Girshick, R.B.: Segment anything. In: ICCV (2023)
25. Larroche, C.: Inductive lateral movement detection in enterprise computer networks. In: ESANN (2024)
26. Lee, W., McCormick, T.H., Neil, J., Sodja, C., Cui, Y.: Anomaly detection in large-scale networks with latent space models. *Technometrics* **64**(2), 241–252 (2022)
27. Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.t., Rocktäschel, T., et al.: Retrieval-augmented generation for knowledge-intensive NLP tasks. In: NeurIPS (2020)
28. Liu, F., Wen, Y., Wu, Y., Liang, S., Jiang, X., Meng, D.: MLTracer: Malicious logins detection system via graph neural network. In: TrustCom (2020)
29. Liu, H., Li, C., Wu, Q., Lee, Y.J.: Visual instruction tuning. In: NeurIPS (2023)
30. Liu, Q., Stokes, J.W., Mead, R., Burrell, T., Hellen, I., Lambert, J., Marochko, A., Cui, W.: Latte: Large-scale lateral movement detection. In: MILCOM (2018)
31. Mao, H., Chen, Z., Tang, W., Zhao, J., Ma, Y., Zhao, T., Shah, N., Galkin, M., Tang, J.: Position: Graph foundation models are already here. In: ICML (2024)
32. Neil, J., Hash, C., Brugh, A., Fisk, M., Storlie, C.B.: Scan statistics for the online detection of locally anomalous subgraphs. *Technometrics* **55**(4), 403–414 (2013)
33. Neil, J., Uphoff, B., Hash, C., Storlie, C.: Towards improved detection of attackers in computer networks: New edges, fast updating, and host agents. In: ISRCS (2013)
34. Oquab, M., Darcet, T., Moutakanni, T., Vo, H., Szafraniec, M., Khalidov, V., Fernandez, P., Haziza, D., Massa, F., El-Nouby, A., Assran, M., Ballas, N., Galuba, W., Howes, R., Huang, P., Li, S., Misra, I., Rabbat, M.G., Sharma, V., Synnaeve, G., Xu, H., Jégou, H., Mairal, J., Labatut, P., Joulin, A., Bojanowski, P.: DINOv2: Learning robust visual features without supervision. arXiv preprint arXiv:2304.07193 (2023)
35. Paudel, R., Huang, H.H.: Pikachu: Temporal walk based dynamic graph embedding for network anomaly detection. In: NOMS (2022)
36. Qiao, H., Niu, C., Chen, L., Pang, G.: AnomalyGFM: Graph foundation model for zero/few-shot anomaly detection. arXiv preprint arXiv:2502.09254 (2025)
37. Sanna Passino, F., Turcotte, M.J., Heard, N.A.: Graph link prediction in computer networks using Poisson matrix factorisation. *Ann. Appl. Stat.* **16**(3), 1313–1332 (2022)
38. Shen, Y., Bevilacqua, B., Robinson, J., Kanatsoulis, C., Leskovec, J., Ribeiro, B.: Zero-shot generalization of GNNs over distinct attribute domains. In: ICML Workshops (2024)
39. Siadati, H., Memon, N.: Detecting structurally anomalous logins within enterprise networks. In: CCS (2017)
40. Sun, X., Yang, J.: HetGLM: Lateral movement detection by discovering anomalous links with heterogeneous graph neural network. In: IPCCC (2022)
41. Tang, B., Hu, Q., Lin, D.: Reducing false positives of user-to-entity first-access alerts for user behavior analytics. In: ICDM Workshops (2017)
42. Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al.: Llama: Open and efficient foundation language models. arXiv preprint arXiv:2302.13971 (2023)

43. Turcotte, M., Heard, N., Neil, J.: Detecting localised anomalous behaviour in a computer network. In: IDA (2014)
44. Wei, R., Cai, L., Yu, A., Meng, D.: AGE: Authentication graph embedding for detecting anomalous login activities. In: ICICS (2019)
45. Xu, J., Shu, X., Li, Z.: Understanding and bridging the gap between unsupervised network representation learning and security analytics. In: SP (2024)
46. Zhang, Y., Bevilacqua, B., Galkin, M., Ribeiro, B.: TRIX: A more expressive model for zero-shot domain transfer in knowledge graphs. In: LoG (2024)
47. Zhang, Z., Jia, J., Wang, B., Gong, N.Z.: Backdoor attacks to graph neural networks. In: SACMAT (2021)
48. Zhang, Z., Liu, Q., Huang, Z., Wang, H., Lee, C.K., Chen, E.: Model inversion attacks against graph neural networks. *IEEE Trans. Knowl. Data Eng.* **35**(9), 8729–8741 (2022)
49. Zhao, J., Mostafa, H., Galkin, M., Bronstein, M.M., Zhu, Z., Tang, J.: Graphany: A foundation model for node classification on any graph. arXiv preprint arXiv:2405.20445 (2024)
50. Zhao, S., Wei, R., Cai, L., Yu, A., Meng, D.: CTLMD: Continuous-temporal lateral movement detection using graph embedding. In: ICICS (2019)
51. Zhu, Z., Zhang, Z., Xhonneux, L.A.C., Tang, J.: Neural Bellman-Ford networks: A general graph neural network framework for link prediction. In: NeurIPS (2021)
52. Zügner, D., Akbarnejad, A., Günnemann, S.: Adversarial attacks on neural networks for graph data. In: KDD (2018)

## A Graph-Based Score Refinement Algorithm

The anomaly score refinement algorithm introduced in Section 3.4 is more formally described below (Algorithm 1).

---

**Algorithm 1:** Graph-based anomaly score refinement.

---

**Data:** Graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{R})$ , score map  $s : \mathcal{E} \rightarrow \mathbb{R}$ .  
**Result:** Refined score map  $s$ .  
**foreach**  $(u, r, v) \in \mathcal{E}$  **do**  
     $s_{\max} = \max \{s[(u', r', v')]; (u', r', v') \in \mathcal{E} \setminus \{(u, r, v)\}, \{u, v\} \cap \{u', v'\} \neq \emptyset\}$ ;  
    **if**  $s[(u, r, v)] > s_{\max}$  **then**  
         $s[(u, r, v)] = s_{\max}$ ;  
    **end**  
**end**  
**return**  $s$

---

## B Additional Details on Dataset Preprocessing

As mentioned in Section 4.1, we make different preprocessing choices than the authors of ARGUS [45]. We present and justify these choices for the OpTC and LANL datasets in Sections B.1 and B.2, respectively.

### B.1 OpTC Dataset

We use the flow start events in the host logs to reconstruct internal network flows. Only flows between internal hosts are included. This yields flows between IP addresses, and we replace these IP addresses with hostnames whenever possible. Specifically, when a flow start event on a host corresponds to a flow initiated from this host, we can associate the source IP of the flow to the name of the host which logged the event. However, since the host logs of only half of the hosts are present in the OpTC dataset, we cannot resolve all IP addresses. We thus leave the unresolved IP addresses as is. Regarding authentication events, we collect all authentication-related events from all available hosts. These include both local and remote log-ins, as well as privileges being granted to new sessions. Accounts related to Desktop Window Manager and User Mode Driver Framework system services (named DWM-\* and UMFd-\*, respectively) are excluded as they are generated on the fly. The main difference between our preprocessing and that of the ARGUS paper [45] lies in the labeling of lateral movement edges. Xu et al. [45] use the same labeling strategy as Paudel and Huang [35], who label all flows generated by compromised hosts after any red team event as lateral movements. This excessively wide definition generates 21,731 lateral movement edges. To make the evaluation more realistic, we manually labeled flow start events that could be traced back to actual lateral movements documented in the ground truth description of red team activity. With this approach, only 626 edges are labeled as lateral movements.

### B.2 LANL Dataset

We use the `auth.txt` and `flows.txt` files of the LANL dataset as sources for authentication events and internal network flows, respectively. The `redteam.txt` file provides precise labels for lateral movement edges. Our preprocessing differs from that of Xu et al. [45] in two key aspects. First, Xu et al. only consider log-ons using the NTLM authentication package, arguing that other events are unrelated to user authentication. This is factually incorrect, as many authentications are performed using other authentication packages (such as Kerberos). In addition, lateral movements in the LANL dataset happen to coincide with NTLM authentication events. Excluding all other authentication packages thus makes lateral movement detection easier, leading to overestimated detection performance. We correct this mistake by including all log-on events into the dataset. The second difference is the temporal scope of the evaluation: while Xu et al. only consider the first 14 days, we include all 58 days. Since all red team events are located in the first 30 days, restraining the scope to the first 14 days mechanically reduces the proportion of benign edges in the dataset. Our preprocessing thus leads to a harder, more realistic evaluation.