Jbeil: Temporal Graph-Based Inductive Learning to Infer Lateral Movement in Evolving Enterprise Networks

Joseph Khoury*, Đorđe Klisura*, Hadi Zanddizari†, Gonzalo De La Torre Parra‡,
Peyman Najafirad§, Elias Bou-Harb*
*Division of Computer Science and Engineering, Louisiana State University, LA, USA
†The Cyber Center for Security and Analytics, The University of Texas at San Antonio, TX, USA
‡The University of the Incarnate Word, TX, USA
§Secure AI and Autonomy Lab, The University of Texas at San Antonio, TX, USA
*jkhour5@lsu.edu, *dklisu1@lsu.edu, †hadi.zanddizari@utsa.edu, ‡gdparra@uiwtx.edu,
§peyman.najafirad@utsa.edu, *ebouharb@lsu.edu

2024 IEEE Symposium on Security and Privacy (SP) | 979-8-3503-3130-1/24/$31.00 ©2024 IEEE | DOI: 10.1109/SP54263.2024.00009

***Abstract*—Lateral Movement (LM) is one of the core stages of advanced persistent threats which continues to compromise the security posture of enterprise networks at large. Recent research work have employed Graph Neural Network (GNN) techniques to detect LM in intricate networks. Such approaches employ transductive graph learning, where fixed graphs with full nodes' visibility are employed in the training phase, along with ingesting benign data. These two assumptions in real-world setups *(i)* do not take into consideration the evolving nature of enterprise networks where dynamic features and connectivity prevail among hosts, users, virtualized environments, and applications, and *(ii)* hinder the effectiveness of detecting LM by solely training on normal data, especially given the evasive, stealthy, and benign-like behaviors of contemporary malicious maneuvers. Additionally, *(iii)* complex networks typically do not have the entire visibility of their run-time network processes, and if they do, they often fall short in dynamically tracking LM due to latency issues with passive data analysis.**

To this end, this paper proposes Jbeil, a data-driven framework for self-supervised deep learning on evolving networks represented as sequences of authentication timed events. The premise of the work lies in applying an encoder on a continuous-time evolving graph to produce the embedding of the visible graph nodes for each time epoch, and a decoder that leverages these embeddings to perform LM link prediction on unseen nodes. Additionally, we enclose a threat sample augmentation mechanism within Jbeil to ensure a well-informed notion on advanced LM attacks. We evaluate Jbeil using authentication timed events from the Los Alamos network which achieves an AUC score of *99.73%* and a recall score of *99.25%* in predicting LM paths, even when *30%* of the nodes/edges are not present in the training phase. Additionally, we assess different realistic attack scenarios and demonstrate the potential of Jbeil in predicting LM paths with an AUC score of *99%* in its inductive and transductive settings, out performing the state-of-the-art by a significant margin.

***Index Terms*—Lateral Movement, Temporal Graph Neural Networks, Authentication Logs, Evolving Enterprise Networks**

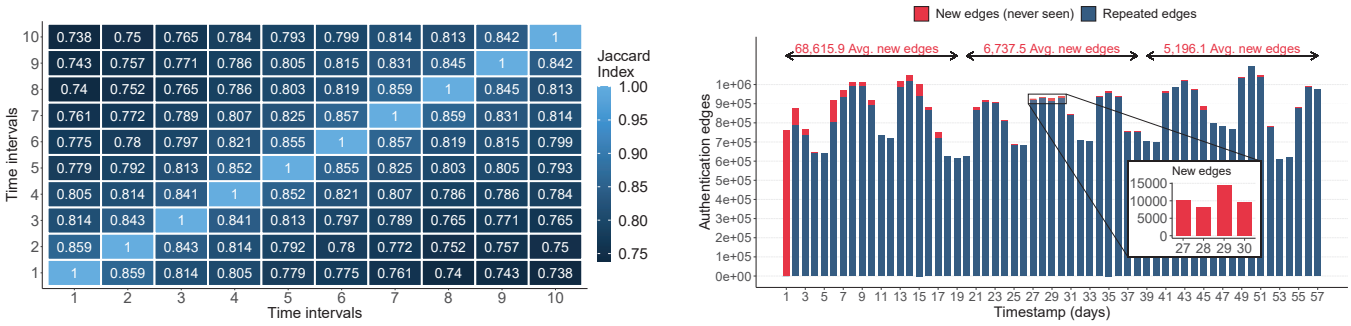
1. Introduction

The distinctive *modus operandi* of Advanced Persistent Threats (APT)s, including their employment of Lateral Movements (LM), typically leaves minimal footprints within both host and network logs. Consequently, key risk indicators would remain undetected by Security Information and Event Management (SIEM) and Network Intrusion Detection Systems (NIDS) used for identifying suspicious events, actions, or trends from broad network and systems' activities. Thus, these detection systems would only be capable of flagging novice LM which are often based on common exploitation frameworks, while failing to provide successful detection measures against LM attempting to exploit newly surfaced vulnerabilities (and related systems) or those employing advanced stealthy procedures.

A plethora of methods for inferring host-, network-, and Internet-based illicit activities (including LM campaigns and pivoting) have been developed over the years. These include detection of anomalies via attack graphs [1]–[8], passive and/or active measurement techniques [9]–[17], machine learning methods [18], deep learning models [19]–[21], game theoretic approaches [22]–[24], heuristic procedures [25]–[29], and graph-based learning approaches [30]–[37]. Indeed, research on the latter methods has been rapidly becoming more mature and is being adopted in a variety of fields, including cyber forensics.

Motivation. To effectively infer LM in enterprise networks, we aim in this paper to devise and develop a temporal graph-based inductive learning approach that embodies two key notions encountered in real-world scenarios, namely, ***(i)*** the evolving/dynamic nature of large and intricate enterprise networks, and ***(ii)*** the pragmatic cyber threat capabilities of these environments. To motivate the notion of evolution in enterprise networks and hence demonstrate the need for effective capabilities which draw-upon network- and systems'-wide artifacts for LM detection, we provide herein (as an example) an empirical analysis of a real-world scenario. As such, we study the

DOI 10.1109/SP54263.2024.00009



(a) **Evolution of nodes**: Jaccard similarity matrix showing the drop of node similarities over time in the network; an approximate *26%* drop in node similarity between the first and last time interval.

(b) **Evolution of edges**: Depicting the number of new edges (never seen before) over time. An average of *5,196.1* new edges emerged between days 39 and 57.

Figure 1: An example study of the LANL computer network's evolution over 58 days demonstrating the emergence of new nodes and connected links.

dynamic nature of nodes and edges obtained from a labeled dataset of authentication events collected from a real-world computer network at the Los Alamos National Laboratory (LANL) spanning 58 days. The nodes represent different users, computers, and servers, while the edges denote the authentication events among the nodes over time. First, at 10 different time intervals (i.e., 58 days equally divided) we measure nodes' similarity at each interval using the Jaccard similarity index described as $J(A,B) = \frac{|A \cap B|}{|A \cup B|}$, where sets A and B represent nodes for two consecutive days. Figure 1a empirically visualises the drop in nodes' similarities by *26%* (between the first and the last time interval), which indeed showcases the evolving nature of enterprise networks in terms of new nodes/authenticating entities. Alternately, Figure 1b illustrates a significant number of new (never seen before) edges over time. Herein, on day 30 of the curated data, *10K* new edges (authentication events) have emerged. This evidence sheds light on the evolving nature of enterprise networks, which would enable malicious actors to infiltrate and laterally maneuver covert nodes and related edges. Additionally, it highlights the difficulty in establishing a sound vantage point for curating and analyzing relevant empirical artifacts from dynamically-maneuvered pathways, further adversely affecting LM detection.

Challenges. Graph Neural Networks (GNNs) have been previously used in host- and network-based anomaly detectors to learn models of benign behaviors and to infer deviations from such models [30]–[35]. Additionally, discrete-time dynamic information was taken into account to improve the detection of anomalous activity in a network while adopting *transductive* learning approaches [31], [32]. However, little scrutiny has been given to *continuous-time dynamic* graph representations and *inductive* learning in security problems (e.g., inferring LM attacks). Furthermore, very limited work have addressed *sampling bias*, *test snooping*, and *temporal snooping* problems (as described in [38]) when implementing machine learning models in computer security contexts. The evolving nature of real and complex enterprise networks (as previously highlighted) further exacerbates these challenges. In this context, `Jbeil`[1] is designed to overcome these grand challenges to effectively infer LM in enterprise networks.

❶ *Continuous-time dynamic graphs* model dynamic graphs with continuous representations, offering superior temporal granularity. Such property circumvents the problem of temporal snooping (i.e., a security-related problem [38]–[40]), and is much needed for correctly representing the underlying distribution under continuous changes found in real-world enterprise networks, where the detection of LM attacks is of paramount importance. Continuous-time dynamic graphs drastically differ from discrete-time dynamic graphs, where the latter consist of sequences of static graph snapshots taken at different intervals in time.

❷ *Inductive learning* in temporal GNNs relates to the key scenario in which the nodes have not been previously seen by the model during training but are used for testing. Unlike transductive learning (see §2), the model must be able to generalize by learning patterns and relationships in the seen data to make accurate predictions on the unseen data. As such, inductiveness jointly contributes to the elimination of test snooping when the test set is used for experiments before the final evaluation [38], as well as in the context of evolving enterprise networks where new nodes and edges continue to dynamically emerge.

❸ *Threat sample augmentation* in APT-related attack scenarios, similar to LM, is crucial to the training stage of the model for addressing the sampling bias problem which occurs whenever the data is not very representative [38]. Such augmentation introduces comprehensive and diverse attackers' Tactics, Techniques, and Procedures (TTPs) (e.g., stealthy or aggressive behaviors, full or limited network knowledge, etc.) to the temporal GNN

1. An old Phoenician city affiliated with the creation and spread of the modern alphabet. Though situated on the coast, literally refers to the mountains—A parallel to the proposed approach being innovative and possessing thorough visibility.

model, hence supporting effective and well-informed decision-making when tested in practice/operations against advanced attack strategies. Nonetheless, curating such exhaustive representative attack data is known to be extremely difficult [36], [38], [41], and hence auxiliary threat sample augmentation methodologies ought to be explored, employed and evaluated.

Contribution. We model a given enterprise network using a graph of authenticating entities, where the main objective is to detect anomalous authentication events among such entities. Such events construct LM paths within the network. where a LM path is characterized by two fundamental properties: ***(i)*** the attacker leverages a new set of credentials, while ***(ii)*** ultimately compromising a network entity that the original actor could not previously access [25]. Such characteristics are essential for an attacker to traverse the network to eventually execute their intended malicious objectives (e.g., stealing or exfiltrating sensitive data). These behaviors are quite challenging as they might resemble legitimate authentication channels with valid credentials in contrast to other noisy attack events. Further, it is imperative to treat authentication events as relational data, where the global context along with its temporal sequence play a vital role in interpreting its behavior. Along this vein, `Jbeil` creates an authentication graph, where nodes represent authenticating entities (such as hosts, users, virtualized environments, and applications) and edges define authentication events connecting these entities. Next, we extract graph features associated with the authentication graph to comprehend the dynamic nature of the enterprise network through evolving nodes' connectivity over time. We then utilize a self-supervised temporal node embedding technique where latent representations are computed for each node in the graph at every time using a message passing technique that leverages the extracted features. In other words, `Jbeil` stores the dynamic state of each node which is updated whenever a node is involved in an event, and subsequently, aggregates the memory of the neighboring nodes when computing the temporal embedding of the given model; this is done by processing the edges ordered in time. Finally, a decoder calculates the edge probabilities and performs LM link prediction.

We train `Jbeil` as an inductive learning model using the calculated temporal node embeddings to adapt to the continuous-time dynamic nature of the data. Specifically, the training is performed on a visible portion of the data, which includes both benign and malicious authentication events to learn about both, individual authentication events and authentication behaviors of the network as a whole. We initially evaluate `Jbeil` on the LANL authentication logs' dataset representing a network capture of a real-world enterprise network with *15,610* nodes. Embedded in these events are red teaming activities that serve as a ground truth for LM related activities. Using `Jbeil` in its inductive setting, we were able to achieve an AUC of *99.82%* and a recall of *99.22%*, even when *30%* of the nodes and edges were never previously introduced during the training stage. Additionally, we enclose within the pre-processing stage of `Jbeil` a threat sample augmentation mechanism that ensure a well-informed notions on recent attack scenarios and tactics. We compare `Jbeil` against the state-of-the-art, `Euler` [31], and show the latter's limitations against the generated attacks in contrast to the transductive setting of `Jbeil` with an average drop of AUC equal to approximately *40%*. Overall, the achieved results demonstrate the novel inductive capabilities of `Jbeil` by generalizing to unseen nodes and its ability to learn and detect realistic LM campaigns.

In summary, this paper contributes in the following:

- We propose `Jbeil`, a temporal graph-based inductive learning approach to detect LM in evolving enterprise networks. The premise of `Jbeil` lies in its continuous-time dynamic nature, message passing and sampling, temporal node embeddings, and finally its decoder responsible for the LM link prediction task.
- We introduce within the pre-processing stage of `Jbeil` an algorithm to scrutinize graph maps and node features to semantically represent the dynamic features and connectivity of entities within the network over time. Additionally, we enclose a threat sample augmentation mechanism that ensure a well-informed notions on recent attack scenarios and tactics.
- We conducted a comprehensive training and evaluation of `Jbeil` in its inductive settings, showcasing its unique capability in predicting LM paths. Furthermore, in comparing the transductive setting of `Jbeil` to that of the state-of-the-art approaches, we demonstrated the superior efficacy of `Jbeil` in learning benign and malicious behaviors when inferring novel LM realistic attack scenarios.
- We make `Jbeil` open source [42] and include all required artifacts and related methods for ease of replicability and reproducibility.

The remaining of the paper is organized as follows. In the next section, we provide relevant fundamental concepts. In §3, we elaborate on the methodological approach embedded within `Jbeil`. We present in §4 the various evaluation setups, experiments and results. Then, §5 present related work. Finally, §6 concludes this work and pinpoints a few endeavors which aim at paving the way for future work.

2. Preliminary

2.1. Lateral Movement and Authentication Logs

LM is a set of internal movements performed by threat actors to propagate inside an enterprise network with the objective of compromising valuable assets (e.g., privileged users, servers, data, etc.) [43], [44]. Initially, the threat actor gains a foothold within a network using techniques such as

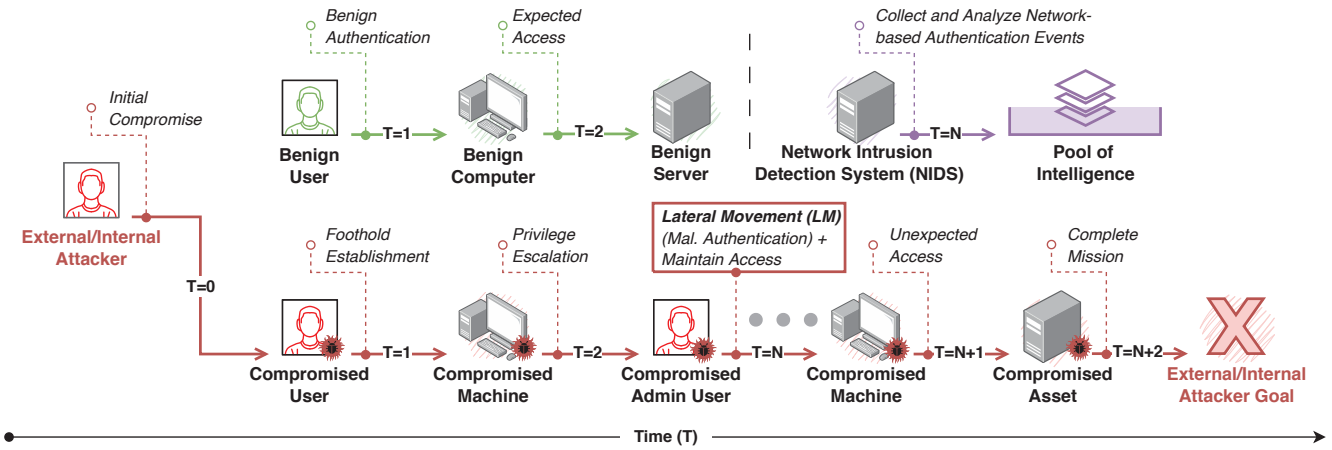


Figure 2: A chronological diagram showcasing the six main stages of an APT attack including LM. Situated at the center of the APT chain, LM is considered one of the main tactics which enables perpetrators to maneuver within the network to broaden their span of access and ultimately complete their illicit objective. LM may occur many times and its detection necessitates careful attention to the spatial and temporal aspects of the events. Our proposed approach `Jbeil` primarily relies on authentication events; one of the primary interface mechanisms among the neighboring and adjacent systems in an enterprise. As such, examining time-stamped authentication events provides an opportunistic vantage point for analyzing malicious events occurring among entities which do not (or rarely) interact.

those identified in the MITRE ATT&CK framework [45]. The subsequent propagation through the network is achieved by exploiting vulnerabilities in existing enterprise entities or authentication protocols [46]. Among the techniques presented in the MITRE ATT&CK framework, specific ones are used for acquiring privileged credentials and unauthorized access during the LM stage [47]–[54]. It is worth noting that LM can be performed manually and/or autonomously (using for instance dedicated malware). Figure 2 presents a time-based overview of the APT attack chain including LM while emphasizing on the crucial role in which time-stamped authentication events exert for inferring LM in enterprise networks. As such, commonly recorded authentication logs generated by various assets in an enterprise network can serve as a (threat) data source to detect suspicious activities that are in close rapport with LM. To curate such activities, network-wide systems such as NIDS and/or SIEM collect and analyze network-based authentication events. Previous research works [31], [32], [55] have leveraged authentication logs for detecting and identifying LM originating from malicious login attempts and unauthorized access. Such methods mainly rely on signature-based or anomaly-based algorithms. While many of these previous methods provide valuable merits for detecting LM in enterprise networks, most of these systems still generate a high volume of false positives and negatives, and/or fail to differentiate between benign and malicious events. To that extent, via `Jbeil`, we aim to explore the trustworthiness of graph-based learning methods for modeling and detecting LM activities using authentication logs.

2.2. Transduction and Induction Reasoning

Link prediction is one of the most important tasks in analyzing graph data [56] given the benefits it can offer in cyber fraud detection, recommendations systems, and knowledge graph completion [57]–[59]. Transductive and inductive methods are two major types of link prediction. The majority of existing work, including [31], [60]–[62], rely on a transductive approach; the adjacency matrix contains all nodes related to the training and test datasets. That is all sets of existing nodes are known and are used during training. In contrast, in the inductive learning approach, all nodes are not required to be known during the training process. Due to such advantage over the transductive approach, the inductive approach allows models to tackle practical problems in production where the model would certainly encounter previously unseen nodes during the training process; in such settings, the only information available to the model would be some attributes of the new nodes used to make a prediction. Figure 3 illustrates both transductive and inductive reasoning where inductive nodes and links are shaded to differentiate them from previously known nodes (i.e., solid nodes and links) in the transductive case. In an evolving enterprise network, one often encounters many unseen (i.e., new nodes) that attempt to communicate with known and/or unknown entities.

Existing GNN models trained within transductive settings are incapable of predicting or classifying new nodes or links. While specific applications may exist where transductive approaches fit very well, many practical applications require inductive approaches to deal with newly integrated nodes. GraphSAGE [63], a recent work addressing inductive link prediction, computes embeddings for unseen nodes

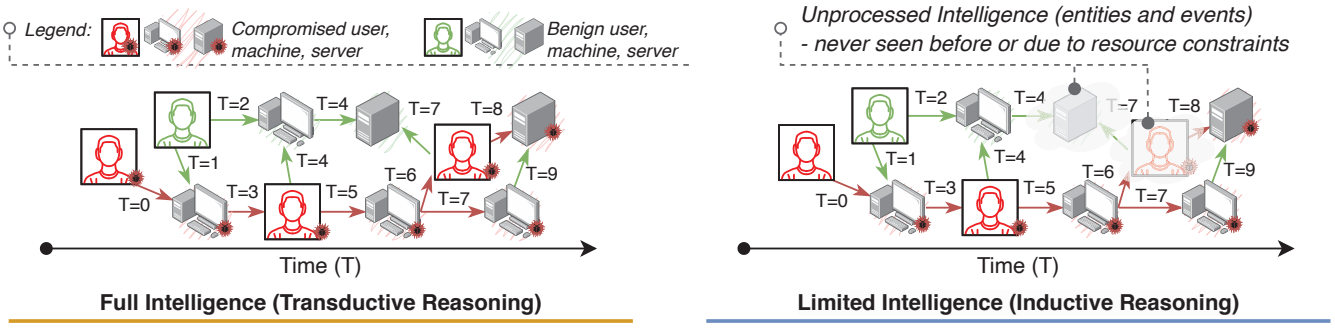


Figure 3: An illustrative comparison between transductive and inductive reasoning in machine learning, specific to a lateral movement scenario inside an enterprise network. In transductive reasoning, all nodes and edges are part of the training process, while in inductive reasoning some nodes and edges are excluded to be later used in the testing stage.

but the information about the nodes' edges is required. In G2G [64], the authors propose an inductive link prediction for unseen nodes without local structures. However, their approach cannot differentiate nodes with common attributes, mainly because their model does not capture the structural information within the nodes' representations. In [65], the authors propose a dual-encoder graph embedding with an alignment for inductive link prediction on unseen nodes where only attribute information is used. Nonetheless, their approach heavily relies on node attributes and does not incorporate the temporal information associated with the interactions among the nodes.

3. Proposed Approach

For modeling and detecting LM behaviors in evolving enterprise networks, we first introduce a pre-processing pipeline for generating a graph structure (derived from the information found in benign and threat augmented authentication logs) by extracting graph maps and calculating graph features. We then discuss `Jbeil`, our proposed temporal graph-based self-supervised inductive learning technique for detecting LM paths. We make `Jbeil` open source on GitHub [42].

3.1. Pre-processing Pipeline

3.1.1. Threat Sample Augmentation. Devising a pragmatic LM detection approach necessitates well-informed notions on recent attack scenarios and tactics. It is also crucial to remediate the sample bias challenge faced in machine learning techniques [38]. However, acquiring real-world LM attack data is known to be extremely difficult [25], [36], [38]. For such reasons, we embed a threat sample augmentation procedure within the pre-processing pipeline of `Jbeil` based on the attack synthesis framework established by Ho et al. [25]. The algorithm used for that purpose is rooted in the breadth-first search (BFS) graph traversal algorithm. Given any enterprise network architecture, we represent the network as a computational graph and utilize the BFS algorithm to parse all the network nodes; enabling the comprehension of the joint spatial, contextual, and temporal topology of the network. This step is important for enabling tailored threat sample augmentation which adheres to the nature and characteristics of the network. The augmentation is performed in two stages. First, random nodes are selected as footholds to initiate the attacks. Second, LM logins are executed by identifying the shortest path to privileged credentials that can access a high-value asset and then the shortest path to the high-value asset using these new credentials. More details on this augmentation scheme, its related generated datasets and experimentation is provided in 4.2.2 and 4.4.2, respectively.

3.1.2. Graph Building and Graph Feature Extraction. Apart from attributes gathered from commonly recorded authentication logs, additional features pertaining to the connectivity dynamics of the network entities are not available. Additionally, the proposed model requires a good graph representation of the network coupled with an effective encoding of its hosts' connectivity. To this end, we make use of host authentication logs to scrutinize the necessary graph features that represent the dynamic nature of the enterprise network as well as the connectivity properties of its hosts and users. The generated graph features are primarily based on the in-degrees and out-degrees of the different hosts and users which are used to support `Jbeil`'s message-passing mechanism among the neighboring hosts. We note that `Jbeil` by default incorporates host-only interaction information; however, additional interaction information pertained to users connectivity is equally needed during the learning phase. To calculate the graph features induced by the different graph interactions (i.e., host-to-host, user-to-host, host-to-user, user-to-user), we follow the approach proposed in [55], [66] to extract the graph map and subsequently calculate the graph features. Figure 4 illustrates the pre-processing pipeline that we follow to extract the graph features and build the graph representations.

In summary, in step ❶, we make use of attributes found in commonly collected authentication logs (and autentication logs associated with our augmented threat sample, see §3.1.1). In step ❷, we extract the graph maps representing

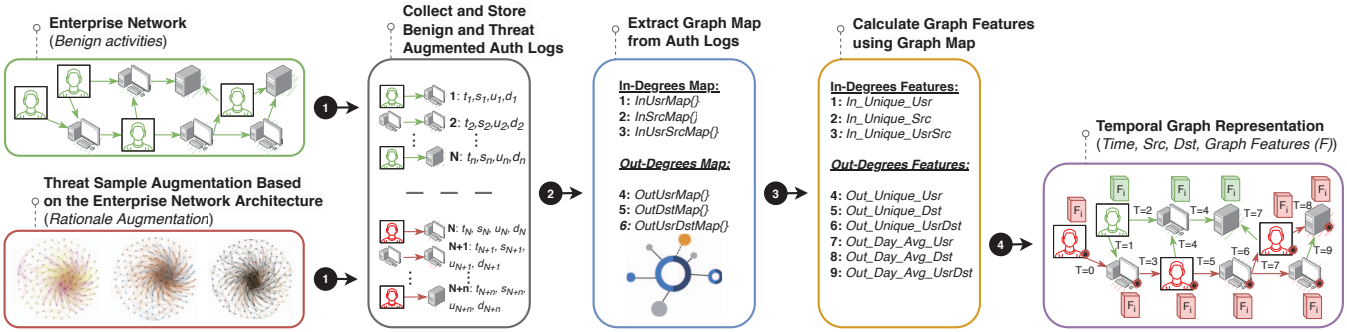


Figure 4: The pre-processing pipeline that depicts the four steps that we follow ***(i)*** leverage and correctly format common authentication logs' attributes (i.e., timestamp, source host, destination host, and user), ***(ii)*** extract the graph map, ***(iii)*** calculate the graph features, and ***(iv)*** aggregate the authentication logs' attributes with the graph features. The outcome is utilized as an input for our proposed approach `Jbeil`.

different interactions within the network. In step ❸, we generate new graph features based on the previously extracted graph maps. Finally, in step ❹, we derive a temporal graph representation capturing time-stamped authentication events among the various nodes and the newly generated graph features. A thorough description of these four distinct steps is presented in the sequel.

Graph map extraction. We initially utilize attributes found in commonly collected authentication logs. Such attributes include timestamp t, source host src, targeted user usr, and destination host dst. This initial step is shown in Figure 4 step ❶. In step ❷ of the pre-processing pipeline, we leverage the previously mentioned authentication log attributes to generate a graph map (i.e., a set of dictionaries) to represent the connectivity between each host and user within the network. We note that calculating the graph maps is essential to preserve users' interaction information since the targeted user attribute usr is omitted in the final graph representation.

After obtaining this initial representation, we extract six different dictionaries, namely: *(i)* in-degree dictionaries, which map the number of incoming authentication events recorded at each host per usr, src, and (usr, src) combination per day (i.e., *InUsrMap*, *InSrcMap*, *InUsrSrcMap*), and *(ii)* out-degree dictionaries, where the number of outgoing daily authentication events recorded at each host per usr, dst, and (usr, dst) combination are mapped (i.e., *OutUsrMap*, *OutDstMap*, *OutUsrDstMap*). To calculate the daily interactions between nodes, we capture calendar dates from the epoch time t. Algorithm 1 presents the $InDegrees$ function to calculate the in-degrees graph map dictionaries, namely, $InUsrMap$, $InSrcMap$, and $InUsrSrcMap$. Specifically, $InDegrees()$ takes as arguments an empty dictionary and its corresponding target x which can be the source Src, the user Usr, and the combination of both Src_Usr. Computing the dictionaries of the out-degrees graph maps, namely, $OutUsrMap$, $OutDstMap$, $OutUsrDstMap$ follows the same approach, but with a small difference where the graph maps at hand should be associated with a source node rather than a destination node.

Algorithm 1: Extract in-degrees graph map from authentication logs.

Require: Authentication logs $AuthLogs$
Ensure: $GraphMap$: $InUsrMap, InSrcMap, InUsrSrcMap$
Ensure: $x : Src, Usr, Src\_Usr$

```
0:  procedure INDEGREES(GraphMap, x)
1:  for all ts, Src, Usr, Dst ∈ AuthLogs do
2:      if Dst ∉ GraphMap then
3:          GraphMap[Dst] ← {}
4:      end if
5:      if Usr ∉ GraphMap[Dst] then
6:          GraphMap[Dst][x] ← {}
7:      end if
8:      day ← ts/86,400
9:      if day ∉ GraphMap[Dst][x] then
10:         GraphMap[Dst][x][day] ← 0
11:     end if
12:     GraphMap[Dst][x][day] ←
        GraphMap[Dst][x][day] + 1
13: end for
13:     return GraphMap
13: end procedure=0
```

The implementations of the graph maps are found on GitHub [42].

Graph feature calculation. In step ❸ of the pipeline, we calculate the graph features using the previously generated dictionaries, which now serve as a graph map representing all interactions within the network. The calculation of the graph features is conducted as follows:

First, we calculate the number of hosts, users, and host-user combinations targeting a specific host using the graph map dictionaries. Particularly, we iterate over all the authentication events to calculate ***(i)*** the number of users, hosts, and user-host combinations per destination host (i.e., In_Unique_Usr, In_Unique_Src, and In_Unique_UsrSrc), and ***(ii)*** the number of users, hosts, and user-host

combination per source host (i.e., Out_Unique_Usr, Out_Unique_Dst, and Out_Unique_UsrDst). Contextually, amid LM, numerous authentication events with ranging frequencies are used by threat actors to propagate within an enterprise network and compromise additional assets. As such, these authentication events may conclude with successful or failed attempts. In either case, such activity will result in an increased number of involved hosts, users, and host-user combinations targeting or being targeted by a specific host. Thus, the newly generated features will append important semantics associated with the connectivity dynamics among hosts and users.

Second, we calculate the daily frequencies of out-degrees pertaining to each host. In essence, we loop over all the authentication events and calculate the daily average interactions of each user, host, and user-host interactions associated with the source host (i.e., Out_Day_Avg_Usr, Out_Day_Avg_Dst, and Out_Day_Avg_UsrDst).

To summarize, calculating these aforementioned features adds valuable semantic meaning to each node as each feature provides additional insights into the node's dynamic behavior while preserving key information associated with users' connectivity dynamics within a network. We highlight in §3.2 the importance of these features in computing the nodes' temporal memories and embeddings.

Temporal graph structure representation. At step ❹ of Figure 4, we combine the previously formatted authentication logs from step ❶ with the calculated features at step ❸, as well as, the label of the authentication event (i.e., malicious or benign) to produce the final graph representation. That said, the graph G will be defined as $G = \{e_{t0}, e_{t1}, e_{t3}, ...\}$, where e represents an authentication event at time t. Each event e comprises the timestamp value of the authentication event t, the source host src, the destination host dst, the label of the event l, and the graph features of source and destination, v_{src} and v_{dst}, respectively. Herein, an event e is defined as follows: $e = \{t, src, dst, l, v_{src}, v_{dst}\}$.

3.2. `Jbeil`: Temporal Graph-Based Inductive Learning to Infer LM

We propose in this work `Jbeil`, a temporal graph-based inductive learning approach rooted in Temporal Graph Networks (TGN) [67] to deal with dynamic graphs represented as sequences of timed authentication events within enterprise networks. Dynamic graphs are primarily characterized by their evolving features and connectivity across time. As previously motivated, enterprise networks are indeed evolving in nature and comprises active nodes consisting of hosts, users, virtualized environments, and applications that are continuously and/or recurrently exchanging authentication events. Coextendingly, threat actors infiltrate and propagate within dynamic enterprise networks to compromise targeted nodes while evading conventional detection techniques. To this end, `Jbeil` uniquely supports continuous-time dynamic graphs represented as a sequence of time-stamped authentication events to calculate the temporal embedding of graph nodes, thereby learning from both temporal and topological data.

3.2.1. Temporal node memory. LM attacks are persistent by nature and remain undetected for an extensive period of time. To this extent, we utilize the memory module of `Jbeil` to retain long-term dependencies for each node in the graph. For every interaction event occurring at time t, we calculate the memory m of each participating interaction node. The calculated memory represents the history of interactions for each node, which is computed every time the node participates in an interaction event. Originally, at *t=0* all the memories are initialized with the node features (recall §3.1.2). For instance, consider Figure 5 which illustrates at ❶ an interaction event at time t where node i is the destination node and nodes j, k, l and o are the source nodes. We note that node o has been previously compromised, which means that its memory at $t-1$ already encompasses information that reflects this situation. As such, the newly calculated memory of node i shown at ❷ will in fact be affected by all its neighboring nodes, including node o. In any interaction event at time t, the memory of the source node and the destination node are calculated using Equations 1 and 2, respectively.

$$m_{src} = msg_{src}(m_{src}(t-1), m_{dst}(t-1), t) \quad (1)$$

$$m_{dst} = msg_{dst}(m_{src}(t-1), m_{dst}(t-1), t) \quad (2)$$

In this work, we use the $msg()$ function as a simple concatenation of the inputs. Additionally, $msg()$ is a learnable function using a recurrent neural network (i.e., Gated Recurrent Unit GRU [68]) and is updated for every event occurring at time t and involving two nodes. To recapitulate, utilizing the memory module in `Jbeil` is important for learning within dynamic enterprise networks due to its ability to store and act-upon long-term information (i.e., a history of interactions) about a node, which is crucial when addressing the LM problem.

3.2.2. Temporal node embedding, inference, and training. Subsequently, we utilize the embedding module of `Jbeil` to calculate the temporal embedding of node i for each time t by aggregating the temporal node memory of all neighboring nodes coupled with its own previously calculated graph features (recall §3.1.2). For example, step ❸ of Figure 5 presents the computation of the final temporal embedding of node i at time t using Equation 3.

$$z_i(t) = \sum_{(0,t)} h(s_{src}(t), s_{dst}(t), v_{src}(t), v_{dst}(t)) \quad (3)$$

We note that $h()$ is a learnable function which may include different formulations such as a simple identity function which uses the memory as a node embedding or an attention function (attention is used in `Jbeil` to ensure adequate visibility) which aggregates information from L-hop temporal neighborhoods [67]. $s(t)$ and $v(t)$ represent the current node memory and the node feature

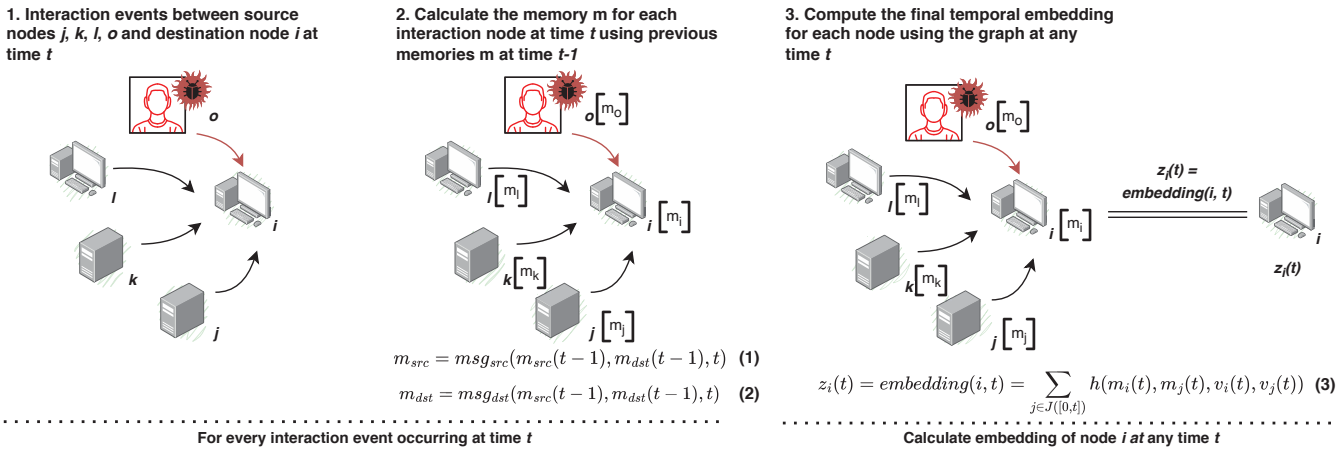


Figure 5: An illustrative representation of an authentication interaction event occurring among node $j, k, l, o,$ *and* i at time t. Node o is a compromised node. Additionally, we showcase in this illustration the temporal node memory and the temporal node embedding modules of `Jbeil` in step *(2)* and step *(3)* respectively. Memory m is calculated for each interaction node using previous memories calculated at $t-1$. The memories are calculated using functions msg_{src} and msg_{dst} for each source node and destination node respectively. Finally, the temporal embedding of node i at time t is calculated by leveraging the memories of the neighboring nodes at t using the function $z_i(t) = embedding(i,t)$.

vector, respectively. As such, calculating the temporal node embedding of i, $z_i(t)$ involves all its temporal neighboring nodes $j \in J$. Computing a node's temporal embedding can solve the staleness problem caused by inactivity, which occurs when a node no longer updates its memory, causing it to become stagnant. This problem occurs when a process remains dormant for a long time or when a user is idle for an extended period. Temporal embedding can mitigate this issue since it depends heavily on the memory and features of neighboring nodes. Finally, after mapping the continuous-time dynamic graph into node embeddings, we leverage `Jbeil`'s decoder which takes one or more node embeddings and perform link prediction by providing the probability of an authentication event (path/edge).

`Jbeil` is trained using a self-supervised approach (i.e., self-supervised link prediction) to detect LM activities using temporal and topological data of both benign and malicious authentication events. In contrast to learning just from normal data (as implemented in state-of-the-art), which limits the inference of novel LM attack scenarios (as demonstrated in §4.2.2), `Jbeil` possesses the unique capability to successfully capture such scenarios using the augmentation method previously discussed (see §3.1.1).

4. Evaluation

Herein, we conduct a thorough experimentation and evaluation of `Jbeil` to demonstrate its effectiveness against various LM attacks within enterprise networks. The objectives are to assess its detection metrics under different LM attack campaigns, validate its inductive capabilities via detection of previously unseen nodes/edges/attacks, while corroborating its superior transductive abilities against a state-of-the-art approach. Additionally, for practically and scalability reasons, its time complexity and inference time is also benchmarked and validated.

4.1. System Setup

We develop, train, and test `Jbeil` in Jetstream Cloud [69] using the TG-CIS200038 allocation. The host system consists of a GPU-based virtual machine of size g3.xl (32 vCPUs, 125GB RAM, 60GB HDD, and 250GB of ephemeral storage) which was deployed with an Ubuntu 20.04.4 image and Docker Engine v20.10.12. The pytorch container image "nvcr.ionvidiapytorch - 22.05-py3" from NGC was used as the development environment during the experimentation process. Torch v1.11.0+cu113 was pre-installed in the container and igraph v0.9.11 was added to support all experiments.

4.2. Datasets

4.2.1. LANL and Pivoting Datasets. In our study, we leverage a de-identified dataset collected from various sources within the Los Alamos National Laboratory's (LANL) [70] corporate internal computer network. The data collection spans over 58 days and includes benign authentication logs and a set of pre-defined red teaming events. The empirical data originates from Windows-based desktop computers and active directory servers. Additionally, we employ the anonymized Pivoting dataset [29] consisting of real network traffic captured by probes situated in a large organization. The network flows are collected over an entire working day and represent communications among internal hosts, as well as labeled flow that are part of a pivoting activity. Such network-based datasets can effectively reveal internal activities of interest that will help in modeling and

Table 1: The Los Alamos National Laboratory's (LANL) corporate internal computer network dataset and the Pivoting dataset captured by probes situated in a large organization.

Dataset	Nodes	Edges	Type	Duration
LANL [70]	15,610	49,341,300	Net. Auth. logs	58 Days
Pivoting [29]	1,015	74,551,643	Network flows	1 Day

detecting LM within large and dynamic enterprise networks. Table 1 provides a brief summary of the employed datasets. Both datasets are transformed into graph representations comprised of *(i)* nodes; representing hosts and users and *(ii)* edges; representing network flows and authentication events among the nodes at a given time.

4.2.2. Augmented Threat Data. As previously described in §3.1.1, we embody in our approach an attack synthesis framework, namely, `lateral-movement-simulator` [25], [71] that is capable of generating a wide variety of real-world attack scenarios and tactics to practically augment `Jbeil` with threat sample data. The framework developed in [25] is designed and implemented for a specific (proprietary) dataset and thus does not port by default to other authentication log datasets. As such, the tool, along with its generated attacks (in [25]), has not been borrowed or directly applied to `Jbeil` and LANL dataset. Particularly, we made a considerable effort to modify its design and implementation, rendering it generic to accept any authentication dataset. Additionally, to generate multiple attacks, we develop two auxiliary methods: the first for producing n number of LM attacks and the second for embedding these attacks into a benign dataset; in our case herein, we use the LANL dataset, which incorporates commonly recorded authentication logs.

To this end, using our modified version of the tool we generate realistic LM attack entries that are specifically tailored to (and in coherence with) the LANL network infrastructure. The generate entries begin at a random date and time to create randomness. We start by randomly selecting nodes in the LANL data as initiating victims, whose machines served as compromised footholds for attackers to launch their LM (i.e., foothold establishment). As such, for each selected victim, we generate different attack scenarios. The generated attack scenarios and their related experimentation are thoroughly elaborated in §4.4.2.

4.3. Evaluation metrics

Optimal threshold using ROC curve. The LANL dataset has skewed data proportions where the number of benign data points excessively exceeds the malicious ones (i.e., LM paths). Therefore, to address this problem, we leverage the ROC curve to find the optimal threshold for our imbalanced data. In essence, we calculate the geometric mean (i.e., G-mean) of sensitivity (i.e., recall) and specificity, which is one of the de-facto unbiased evaluation metrics for imbalanced classification. We use this optimal threshold to classify the prediction scores during the evaluation stage of `Jbeil`.

Precision score. We use this metric to quantify the positive class prediction that belongs to the positive class. As such, the precision score of Jbeil indicates its quality of correctly inferring LM without blocking legitimate authentication. The precision score is defined as follows.
$Precision\ score = True\ Positives/(True\ Positives + False\ Positives)$.

Recall score. We use this metric to measure the count of the positive class that the model can correctly predict over all the actual positive values. In other words, the recall score of Jbeil signifies the percentage of correctly inferring LM without missing any small number of LMs. The recall score is defined as follows.
$Recall\ score = True\ Positives/(True\ Positives + False\ Negatives)$.

Average Precision (AP) score. We use this to compile the precision and recall curve as the weighted mean of precision at each threshold n. The AP score is defined as follows.
$AP = \sum_n (Recall_n - Recall_{n-1}) \times Precision_n$.

Area Under the Curve (AUC) score. We use this to quantify the proportion of true positives in contrast to the proportion of false positives. It is the area under the curve created by plotting the True Positive Rate, $TPR = TP/(TP + FN)$ and the False Positive Rate, $FPR = FP/(FP+TN)$ as the thresholds for classification change.

4.4. Experiments

4.4.1. Induction Reasoning Experimentation. Since `Jbeil` supports inductive and self-supervised learning, we explicitly mask an *X* random sample of nodes with their corresponding connected edges to train only on visible nodes/edges. Contextually, this evaluation attempts to infer/test for (previously) unknown/unseen attacks by only learning on a set of known events. We note that the masking process does not affect the temporal sequence of authentication events within the data. Knowing that enterprise networks are dynamic in nature and continuously involve newly emerged network entities, we conduct herein three experiments which demonstrate the unique inductive capabilities of `Jbeil` in predicting LM paths on unseen nodes/edges. To accomplish this, we randomly mask nodes and edges in the LANL dataset at three different ratios (i.e., *30%*, *40%*, *50%*) to perform the inductiveness testing. We ran the experimentation several times to validate the consistency of the results. Note that the masked nodes are never introduced during the testing stage. Table 2 presents `Jbeil`'s results via the induction reasoning experimentation.

For each experiment, we train `Jbeil` using a specific number of training nodes, then we perform a transductive evaluation by leveraging a test samples with previously seen nodes. Subsequently, using the same trained model we use the masked nodes to conduct an inductive test and thus evaluate the performance of `Jbeil` on unseen nodes. At each

Table 2: Jbeil's inductive reasoning experiments: In Experiment 1, we use *9,886* nodes to train the model and *1,041* for transductive evaluation. We also use (*15,610 - 9,886 - 1,041 = 4,683*) as inductive test nodes (unseen nodes). In Experiments 2 and 3, we decreased the number of the training nodes to obtain more unseen nodes. The obtained results highlight the generalizability of *Jbeil* where it can provide comparable performance to predict seen and unseen nodes, i.e., inductive performance is almost the same as the transductive one.

Experiments	**Experiment 1**		**Experiment 2**		**Experiment 3**	
Training nodes	**9,886**		**8,423**		**6,943**	
Reasoning	**Transductive**	**Inductive**	**Transductive**	**Inductive**	**Transductive**	**Inductive**
Nodes #	1,041	4,683	943	6,244	862	7,805
Precision (%)	98.93	98.48	97.65	97.52	67.85	66.83
Recall (%)	99.22	99.25	91.47	91.23	97.58	97.76
AP (%)	99.80	99.63	93.72	93.49	67.45	66.49
AUC (%)	99.82	99.73	94.76	94.59	75.62	74.55

experiment, we reduce the number of training nodes and correspondingly the transductive test samples, and on the other hand, we increase the number of inductive test samples to demonstrate the capabilities of Jbeil at different levels. Furthermore, we train Jbeil over *10* epochs using a *0.005* learning rate and a patience factor of *5*.

Experiment 1. We randomly mask *30%* of the LANL dataset (*15,610 * 0.3 = 4,683* nodes) as an unseen test samples to evaluate the inductive performance of Jbeil. Referring to Table 2, we train Jbeil using *9,886* nodes and perform a transductive evaluation using previously seen *1,041* nodes. Jbeil achieves an AUC, AP, Recall, and Precision scores of *99.82%*, *99.80%*, *99.22%*, and *98.93%* respectively. These results are in fact very high knowing that the transductive testing nodes were all introduced during training. Now, using *4,683* previously unseen nodes during training, Jbeil achieves an AUC, AP, Recall, and Precision scores of *99.73%*, *99.63%*, *99.25%*, and *98.48%*, respectively. This inductive test results are almost the same as the transductive ones demonstrating the unique capability of Jbeil to generalize on unseen nodes. As such, we conclude from experiment 1 that Jbeil can efficiently learn the time sequentiality and semantics of authentication events and successfully predict LM paths on unseen nodes in evolving enterprise network where *30%* of its nodes were never seen. Such variation in unseen nodes can be contextually associated with unprocessed threat indicators, stealthy 0-day attacks exploiting covert LM paths, lack of broad data visibility, or the inception of new nodes linked to newly emerged hosts/users.

Experiment 2. Subsequently, in experiment 2 we aim to overburden Jbeil by increasing the number of masked nodes to *40%* (*15,610 * 0.4 = 6,244* nodes) and accordingly train Jbeil using only *8,423* nodes. We observe in Table 2 a decline in the transductive test evaluation on *942* samples were Jbeil achieves an AUC score of *94.76%*, with a drop of almost *5%*. Reducing the number of nodes during training have a direct negative impact on the number of neighboring nodes which are essential in computing the node embedding, hence the decay in effectively learning the time sequentality and semantics of both benign and malicious (i.e., LM paths) authentication events. Since we are using the same trained model to perform the inductive evaluation, we observe a similar drop in the test results were the AUC score reached *94.59%* on unseen nodes. Although the results are lower than experiment 1, yet, we can deduce that Jbeil is still robust and generaizable to newly emerged graph nodes and interactions, and hence capable of performing LM link predictions in evolving enterprise networks were *40%* of the network entities are newly emerged.

Experiment 3. In this experiment, we further increase the number of inductive nodes to *50%* (*15,610 * 0.5 = 7,805* nodes) and train Jbeil on only *6,943*. In fact, this experiment falls under a maximal plot where the number of unseen nodes exceeds the number of seen nodes used during training (extreme inductive case). Nonetheless, this case can help us understand the utmost capabilities of Jbeil and also provide us with valuable recommendations and guidance on the adequate intelligence needed to achieve successful LM path predictions in evolving enterprise networks. Table 2 shows an AUC score of *75.62%* on *862* visible nodes and an AUC score of *74.55%* on *7,805* masked nodes. In this particular case, the number of training nodes and edges is significantly smaller than that of the testing (masked) ones. As a result, message passing, aggregation, and sampling in Jbeil are negatively impacted, ultimately affecting the calculated temporal node embedding. Specifically, these embeddings are unable to capture sufficient knowledge from their spatial and temporal neighborhoods (connected nodes and linked edges) due to the limited number of neighboring nodes present during training. However, during the testing stage, the previously masked nodes and edges become unmasked, thereby adding substantial information to the graph that Jbeil did not encounter during training. This discrepancy between the training and testing datasets ultimately impacts the performance of Jbeil in predicting the outcomes of new samples. Nonetheless, this experiments clearly highlight the maximal limit of Jbeil while providing unique and practical insights on the training and testing strategies of our approach.

Table 3: Jbeil's generalizability: An inductive and transductive experiment on the Pivoting dataset [29].

Network Flows	Training nodes	1,015	
	Reasoning	Transductive	Inductive
	Nodes #	609	406
Pivoting dataset	Precision (%)	99.12	99.09
	Recall (%)	97.09	97.42
	AP (%)	97.72	97.84
	AUC (%)	98.12	98.26

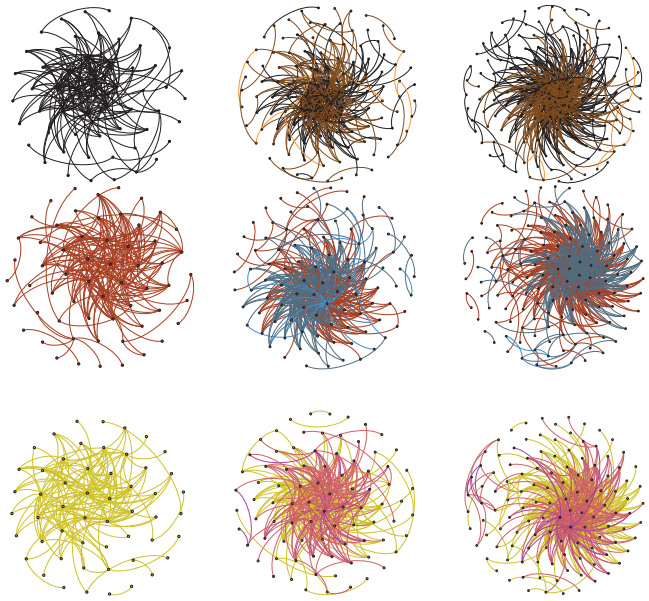
Table 4: Jbeil's results for detecting LM under Attack Scenarios 1, 2 and 3.

Authentication attacks	Training nodes	9,886	
	Reasoning	Transductive	Inductive
	Nodes #	1,041	4,683
LANL - scenario 1	Precision	99.21	99.31
	Recall	98.17	98.84
	AP	99.67	99.49
	AUC	99.66	99.61
LANL - scenario 2	Precision	99.38	99.31
	Recall	98.24	99.08
	AP	99.61	99.44
	AUC	99.54	99.60
LANL - scenario 3	Precison	99.17	99.27
	Recall	97.95	98.83
	AP	99.59	99.36
	AUC	99.54	99.54

To establish the generalizability of our approach, we conducted a comprehensive evaluation of Jbeil based on the Pivoting dataset [29]. We first mask *40%* of the original dataset (*i.e., 1,015*0.4 = 406* sample nodes) to generate an unseen test set and then we train the model on the rest of the data. The results of this evaluation are presented in Table 3, which shows an impressive AUC score of *98.26%* and a precision of *99.09%*, demonstrating the excellent generalizability of Jbeil to different network-based datasets. Moreover, these results underscore the unique inductive capabilities of Jbeil, which can consistently provide high-quality predictions even for nodes that were never encountered during training.

4.4.2. Augmented Threat Sample Experimentation. By introducing the benign LANL dataset (cleaned from the red teaming activities) to our self-modified empirical attack synthesis framework by Ho et al. [25], [71] we specifically generate and instrument a wide variety of real-world LM attacks that conforms to the LANL network system and data distribution. Subsequently, we embed these attacks in the LANL dataset to ultimately achieve a sample augmentation mechanism in Jbeil's pre-processing pipeline.

Within this attack synthesis framework, different parameters are available to define practical LM attacks. These include attackers' stealthiness, goals, and knowledge; we use these to generate distinct and realistic attack scenarios. Specifically, the attack framework generates logins by adhering to the predefined scenario's parameters. For instance, these parameters are used to specify when an attack succeeds and how the attacker maneuvers through the network, while



(a) t_1 (b) t_2 (c) t_3

Figure 6: A depiction of the augmented threat datasets showing lateral movement attacks across time intervals t_1, t_2, t_3 utilizing the Fruchterman-Reingold layout. The first row depicts Scenario 1, the second-row Scenario 2, and the third-row Scenario 3. Nodes colored differently at time t_2 indicate newly compromised nodes from time t_1, and nodes colored differently at time t_3 indicate newly compromised nodes from time t_2.

also maintaining how much the attacker knows about the network. The attack framework generated and integrated *1,801* realistic LM attacks into the LANL dataset. These were organized across three distinct attack scenarios (as below). For each, the LM attack framework chose target nodes at random.

Scenario 1. Here, the attacker only knows about the history of the machines which they have previously maneuvered into, and the attack stops producing new logins once it has acquired access to a system to which its initial victim did not have access to. The attack framework produced *695* of these attacks and incorporated them into the LANL dataset.

Scenario 2. The attacker is knowledgeable of the whole network topology. The attack terminates after accessing 50 devices or logging into every machine accessible with the final credential set. In addition, the attacker only generates logins that traverse edges that valid users have already traversed. The attack framework generated *606* of these attacks and incorporated them into the LANL dataset.

Scenario 3. Here, the attacker is knowledgeable of the whole network topology and performs attacks by multiple logins until access is gained to a server with high value. In this case, an attacker does not only creates logins that go through edges that valid users have already gone through but also uses a set of credentials in a login, only if its authorized user has recently logged in to the source machine.

Table 5: A comparative experiment showing the performance of Euler [31] and GraphSAGE [72] in detecting LM using authentication logs across the LANL dataset (red teaming and the generated three attack scenarios). Our proposed model Jbeil is superior across all the attacks.

Net. Auth. Attacks	Reasoning Approach	Transductive					Inductive	
		Euler GCN-GRU	Euler GCN-LSTM	Euler SAGE-LSTM	Euler GAT-None	Jbeil	GraphSAGE	Jbeil
LANL - redteam	AUC	96.96	98.49	95.95	91.47	99.31	76.77	99.73
LANL - Scenario 1	AUC	58.60	41.67	36.87	47.37	98.59	69.86	99.61
LANL - Scenario 2	AUC	59.96	58.68	55.70	51.98	98.76	72.54	99.60
LANL - Scenario 3	AUC	63.84	66.63	60.29	58.44	98.53	72.14	99.54

The attack framework produced *500* of these attacks and incorporated them into the LANL dataset. We split such obtained dataset (of the three attack scenarios) into three equal parts to show how the LM attacks have evolved over time. Figure 6 demonstrates the progression of the generated LM attacks throughout time. In Scenario 1, the number of nodes at time t_1 comprises *55*, at time t_2 involves *115*, and at time t_3 encompasses *160*. In Scenario 2, the number of nodes is *59* at time t_1, *101* at time t_2, and *151* at time t_3. In Scenario 3, the number of nodes at time t_1, t_2, t_3 is *53*, *85*, and *120*, respectively.

After merging the obtained malicious data with the original LANL dataset, we train Jbeil to evaluate its ability to detect these new threats. Table 4 summarizes the results of Jbeil when applied to the LANL dataset in order to infer the newly embedded LM attacks of the three different scenarios. We conduct three experiments pertained to the three different attack scenarios; training Jbeil on *9,886* visible nodes while performing transductive testing on *1,041* visible test samples. As such Jbeil successfully detects the three scenarios with AUC scores of *99.66%*, *99.54%*, and *99.54%*, respectively. Additionally, we test Jbeil on *4,683* inductive nodes (which have never been seen during the training phase). Similar to the results achieved on the seen nodes, Jbeil successfully detects LM paths associated with attack scenarios 1, 2, and 3 with AUC scores of *99.61%*, *99.60%*, and *99.54%*, respectively.

4.4.3. Jbeil's Efficiency and Scalability: *inductive and transductive training and inference benchmark*. Jbeil was designed with practicality and scalability in mind. To validate such an assumption, we benchmarked its time comlexity during both inductive and transductive training under various networks. Specifically, we induce multiple subgraphs using the LANL dataset with different number of nodes and edges and evaluate the time required for Jbeil to train upon. Figure 7 shows that Jbeil's training time complexity is highly correlated with the number of edges ans nodes. For instance, in the transductive case when the number of nodes is *14.9K* and the number of edges is *34M* the time complexity is very high; reaching *5,054 seconds* with memory usage of *6GB*. This clearly exhibits the implications of transductive reasoning when faced with significant number of nodes and edges in large enterprise network. On the other hand, in an inductive case where part of the nodes and edges are masked, for instance when the number of nodes is *9.9K* and the number of edges is *2.1M*,

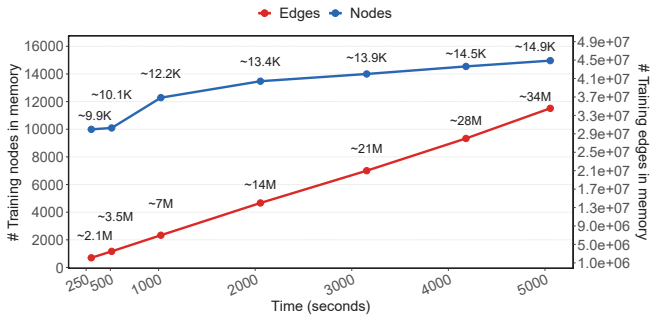


Figure 7: Time complexity of Jbeil on different subgraphs/subnets with a various number of nodes and edges.

the time complexity is much less, scoring *303 seconds* with a memory usage of *1.5GB*. This reflects on the pivotal importance of Jbeil's inductive reasoning in ensuring perfect results (recall Tables 2 and 4) while guarantying scalability and efficiency.

After conducting numerous experiments, we have observed that the LM inference time for Jbeil varies depending on the size of the graph (nodes and edge). During evaluation of both seen and unseen nodes and edges, we have found that the inference time ranges from *12 seconds* for a graph with *1 million* edges, to *5 minutes and 45 seconds* for a graph with *30 million* edges. To this end, we note that Jbeil in-light of large and dynamic networks is scalable and efficient when computing resources are minimal, training and response time is bounded, and/or intelligence processing is sparse.

4.4.4. Jbeil's Comparisons against State-Of-The-Art. Firstly, Jbeil is contrasted with Euler [31], the state-of-the-art transductive graph-based approach for LM detection. Initially, we obtained Euler's source code and successfully compiled four different variants of it to test its efficacy in detecting LM paths. In these experiments, we evaluate Euler with the LANL red team attacks, as well as the three attack scenarios that the attack framework have produced. Table 5 summarizes the results of Euler's variations [31] in detecting such attacks. sEuler performed quite well in detecting the red team attacks with an AUC of *96.96%* (Euler's GCN-GRU variation); this is the exact same result that the authors have previously reported under such settings. Nevertheless, Euler failed to detect the three generated LM scenarios with scores as low as *36.87%* for Scenario 1 and

Table 6: An ablation study on the temporal representation of edges in Jbeil.

Approach	Time Representation	Performance (AUC %)
Jbeil	**Continuous Time**	99.31
Jbeil (modified)	**Static Time**	69.92
Euler	**Discrete Time**	98.49

55.70% for Scenario 2. Jbeil outperformed Euler across the entire set of attacks. We attribute Euler's low detection rate when executed on the three LM attack scenarios due to the fact that it only learns on a fixed discrete time-dynamic graph (i.e., snapshots of graphs) using benign data. Thus, the model does not fully comprehend the semantics of the attacks. In addition, Euler does not add entity features but solely looks at the node and its interactions.

Secondly, we conducted a comparative analysis with GraphSAGE [72], a well-known method for inductive reasoning in large graphs. Although GraphSAGE is not tailored for LM detection and cybersecurity, we made use of PyG's GraphSAGE implementation [73], [74], which allowed us to fully implement it and perform a comprehensive comparison. We found that while GraphSAGE has proven effective for graph-based machine learning, its lack of capability to incorporate temporal aspects or account for the continuous time-dynamic nature of graphs, along with the absence of cybersecurity data augmentation, pose significant challenges in practical security applications such as LM detection. These limitations are reflected in the results presented in Table 5, which demonstrate GraphSAGE's inferior performance in the four different scenarios in contrast to the superior performance of Jbeil.

Differently, one of the most significant contributions of Jbeil lies in its innovative continuous temporal representation. In comparison to alternative temporal representations such as static time or discrete time-dynamic, Jbeil's continuous time-dynamic capabilities have proven to be especially advantageous. To evaluate the effectiveness of Jbeil's temporal representation, we conducted experiments to compare the performance of a variant of Jbeil without the temporal aspect. Our results, presented in Table 6, demonstrate the clear superiority of Jbeil's continuous time-dynamic representation over static time representation. Furthermore, we have found that Jbeil outperforms Euler's discrete time-dynamic approach, further supporting the strength of Jbeil's temporal representation.

4.5. Jbeil's Potential Real-World Integration

The task of inferring LM in large enterprise networks is complex and challenging for existing SIEM and Network Detection and Response (NDR) systems. However, integrating Jbeil into these systems can enhance security data through advanced LM detection, leading to better collaboration, automation, threat analysis, incident response, and mitigation capabilities. This can be achieved using modules such as Google Cloud Pub/Sub [75], Syslog, or Logstash [76] to collect and relay Jbeil-generated security logs to other third-party security solutions.

5. Related Work

Existing graph-based methods for LM and APT detection. Recently, King and Huang proposed ,Euler [31], a state-of-the-art discrete-time dynamic graph neural network that enables scalable dynamic link prediction. Euler was tested with the LANL dataset with the objective of detecting lateral movement during APT campaigns. Other works, such as ShadeWatcher [33], Depcomm [34], Threatrace [35], and APT-KGL [36] employed graph-based methods including graph-based recommendation systems, graph summarizing, GraphSAGE, and heterogeneous provenance graphs in an effort to assist host-based APT summarizing, detection, and tracing.

Graph-related advancements, challenges, and opportunities. Significant progress has been made in addressing the challenges associated with using graphs for node classification, link prediction, and clustering. Important contributions in this field include: *(i)* node2vec [77] by Grover and Leskovec, which presents an algorithmic framework for learning continuous feature representations for nodes in networks, *(ii)* Deepwalk by Perozzi et al. [78], which provides a scalable approach for learning latent representations of vertices in a network, and *(iii)* Graph Factorization (GF) by Ahmed et al. [79], which proposes a framework for large-scale graph decomposition and inference that partitions a graph over multiple systems where each partition reduces the number of neighboring nodes. Despite the promising results and advancements, graph-based techniques used in cyber security still have limitations, including constraints in reasoning and inadequate representation of dynamic environments, which fail to accurately model large enterprise networks. Our approach, Jbeil, uses TGN, a deep learning framework that operates on continuous-time dynamic graphs and ensures generalizability on unseen data points using inductive-based learning.

6. Conclusion

In this work, we propose Jbeil, a temporal graph-based approach for detecting LM in enterprise networks using time-stamped authentication logs. Jbeil's encoder aggregates neighboring node memories to compute temporal node embeddings, which are used by the decoder for LM link prediction. Our experiments show Jbeil's inductive capabilities in predicting LM paths with an AUC score of *99.73%* when *30%* of the dataset is masked. Jbeil also detects LM paths regardless of node visibility and quantity and outperforms state-of-the-art work in predicting LM paths for realistic attacks. We solidly proved the efficiency and scalability of Jbeil and factually discussed its potential integration with existing security solutions. In future work, we plan on exploring Jbeil's potential for detecting other types of attacks and malicious activities by analyzing host-based data and logs.

Acknowledgments

The authors would like to thank the Program Committee, anonymous reviewers, and the shepherd for their valuable feedback and suggestions that significantly improved the quality of this paper. We also acknowledge funding from the National Science Foundation (NSF) under grant numbers #2230086 and #2219772.

References

[1] M. Albanese, S. Jajodia, and S. Noel, "Time-efficient and cost-effective network hardening using attack graphs," in *IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2012)*. IEEE, 2012, pp. 1–12.

[2] S. Jha, O. Sheyner, and J. Wing, "Two formal analyses of attack graphs," in *Proceedings 15th IEEE Computer Security Foundations Workshop. CSFW-15*. IEEE, 2002, pp. 49–63.

[3] X. Ou, W. F. Boyer, and M. A. McQueen, "A scalable approach to attack graph generation," in *Proceedings of the 13th ACM conference on Computer and communications security*, 2006, pp. 336–345.

[4] J. Homer, X. Ou, and M. A. McQueen, "From attack graphs to automated configuration management—an iterative approach," *Kansas State University Technical Report*, 2008.

[5] R. E. Sawilla and X. Ou, "Identifying critical attack assets in dependency attack graphs," in *European Symposium on Research in Computer Security*. Springer, 2008, pp. 18–34.

[6] H. Huang, S. Zhang, X. Ou, A. Prakash, and K. Sakallah, "Distilling critical attack graph surface iteratively through minimum-cost sat solving," in *Proceedings of the 27th Annual Computer Security Applications Conference*, 2011, pp. 31–40.

[7] J. Lee, H. Lee, and H. P. In, "Scalable attack graph for risk assessment," in *2009 International Conference on Information Networking*. IEEE, 2009, pp. 1–5.

[8] B. M. Bowen, S. Hershkop, A. D. Keromytis, and S. J. Stolfo, "Baiting inside attackers using decoy documents," in *International Conference on Security and Privacy in Communication Systems*. Springer, 2009, pp. 51–70.

[9] E. Bou-Harb, M. Debbabi, and C. Assi, "Cyber scanning: a comprehensive survey," *Ieee communications surveys & tutorials*, vol. 16, no. 3, pp. 1496–1519, 2013.

[10] ——, "On fingerprinting probing activities," *computers & security*, vol. 43, pp. 35–48, 2014.

[11] ——, "A statistical approach for fingerprinting probing activities," in *2013 International Conference on Availability, Reliability and Security*. IEEE, 2013, pp. 21–30.

[12] C. Fachkha, E. Bou-Harb, and M. Debbabi, "Inferring distributed reflection denial of service attacks from darknet," *Computer Communications*, vol. 62, pp. 59–71, 2015.

[13] E. Bou-Harb, N.-E. Lakhdari, H. Binsalleeh, and M. Debbabi, "Multidimensional investigation of source port 0 probing," *Digital Investigation*, vol. 11, pp. S114–S123, 2014.

[14] C. Fachkha, E. Bou-Harb, A. Keliris, N. D. Memon, and M. Ahamad, "Internet-scale probing of cps: Inference, characterization and orchestration analysis." in *NDSS*, 2017.

[15] J. Khoury, M. Safaei Pour, and E. Bou-Harb, "A near real-time scheme for collecting and analyzing iot malware artifacts at scale," in *Proceedings of the 17th International Conference on Availability, Reliability and Security*, 2022, pp. 1–11.

[16] M. S. Pour, J. Khoury, and E. Bou-Harb, "Honeycomb: A darknet-centric proactive deception technique for curating iot malware forensic artifacts," in *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2022, pp. 1–9.

[17] V. Rammouz, J. Khoury, . Klisura, M. S. Pour, M. S. Pour, C. Fachkha, and E. Bou-Harb, "Helium-based iot devices: Threat analysis and internet-scale exploitations," in *2023 19th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. IEEE, 2023, pp. 206–211.

[18] T. Bai, H. Bian, M. A. Salahuddin, A. Abou Daya, N. Limam, and R. Boutaba, "Rdp-based lateral movement detection using machine learning," *Computer communications*, vol. 165, pp. 9–19, 2021.

[19] G. D. L. T. Parra, P. Rad, and K.-K. R. Choo, "Implementation of deep packet inspection in smart grids and industrial internet of things: Challenges and opportunities," *Journal of Network and Computer Applications*, vol. 135, pp. 32–46, 2019.

[20] G. D. L. T. Parra, P. Rad, K.-K. R. Choo, and N. Beebe, "Detecting internet of things attacks using distributed deep learning," *Journal of Network and Computer Applications*, vol. 163, p. 102662, 2020.

[21] G. D. L. T. Parra, L. Selvera, J. Khoury, H. Irizarry, E. Bou-Harb, and P. Rad, "Interpretable federated transformer log learning for cloud threat forensics," *Network and Distributed Systems Security (NDSS) Symposium 2022*, 2022.

[22] M. A. Noureddine, A. Fawaz, W. H. Sanders, and T. Başar, "A game-theoretic approach to respond to attacker lateral movement," in *International Conference on Decision and Game Theory for Security*. Springer, 2016, pp. 294–313.

[23] J. Khoury and M. Nassar, "A hybrid game theory and reinforcement learning approach for cyber-physical systems security," in *NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2020, pp. 1–9.

[24] M. Nassar, J. Khoury, A. Erradi, and E. Bou-Harb, "Game theoretical model for cybersecurity risk assessment of industrial control systems," in *2021 11th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*. IEEE, 2021, pp. 1–7.

[25] G. Ho, M. Dhiman, D. Akhawe, V. Paxson, S. Savage, G. M. Voelker, and D. Wagner, "Hopper: Modeling and detecting lateral movement," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 3093–3110.

[26] S. M. Milajerdi, R. Gjomemo, B. Eshete, R. Sekar, and V. Venkatakrishnan, "Holmes: real-time apt detection through correlation of suspicious information flows," in *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2019, pp. 1137–1152.

[27] C. Xiong, T. Zhu, W. Dong, L. Ruan, R. Yang, Y. Chen, Y. Cheng, S. Cheng, and X. Chen, "Conan: A practical real-time apt detection system with high accuracy and efficiency," *IEEE Transactions on Dependable and Secure Computing*, 2020.

[28] X. Han, T. Pasquier, A. Bates, J. Mickens, and M. Seltzer, "Unicorn: Runtime provenance-based detector for advanced persistent threats," *arXiv preprint arXiv:2001.01525*, 2020.

[29] G. Apruzzese, F. Pierazzi, M. Colajanni, and M. Marchetti, "Detection and threat prioritization of pivoting attacks in large networks," *IEEE Transactions on Emerging Topics in Computing*, 2017.

[30] B. Bowman, C. Laprade, Y. Ji, and H. H. Huang, "Detecting lateral movement in enterprise computer networks with unsupervised graph {AI}," in *23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020)*, 2020, pp. 257–268.

[31] I. J. King and H. H. Huang, "Euler: Detecting network lateral movement via scalable temporal link prediction," *Network and Distributed Systems Security (NDSS) Symposium 2022*, 2022.

[32] R. Paudel and H. H. Huang, "Pikachu: Temporal walk based dynamic graph embedding for network anomaly detection," in *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2022, pp. 1–7.

[33] J. Zeng, X. Wang, J. Liu, Y. Chen, Z. Liang, T.-S. Chua, and Z. L. Chua, "Shadewatcher: Recommendation-guided cyber threat analysis using system audit records," in *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, 2022, pp. 1567–1567.

[34] Z. Xu, P. Fang, C. Liu, X. Xiao, Y. Wen, and D. Meng, "Depcomm: Graph summarization on system audit logs for attack investigation," in *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2022, pp. 540–557.

[35] S. Wang, Z. Wang, T. Zhou, H. Sun, X. Yin, D. Han, H. Zhang, X. Shi, and J. Yang, "Threatrace: Detecting and tracing host-based threats in node level through provenance graph learning," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 3972–3987, 2022.

[36] T. Chen, C. Dong, M. Lv, Q. Song, H. Liu, T. Zhu, K. Xu, L. Chen, S. Ji, and Y. Fan, "Apt-kgl: An intelligent apt detection system based on threat knowledge and heterogeneous provenance graph learning," *IEEE Transactions on Dependable and Secure Computing*, 2022.

[37] H. He, Y. Ji, and H. H. Huang, "Illuminati: Towards explaining graph neural networks for cybersecurity analysis," in *2022 IEEE 7th European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2022, pp. 74–89.

[38] D. Arp, E. Quiring, F. Pendlebury, A. Warnecke, F. Pierazzi, C. Wressnegger, L. Cavallaro, and K. Rieck, "Dos and don'ts of machine learning in computer security," in *31st USENIX Security Symposium (USENIX Security 22)*, 2022, pp. 3971–3988.

[39] F. Maggi, W. Robertson, C. Kruegel, and G. Vigna, "Protecting a moving target: Addressing web application concept drift," in *Recent Advances in Intrusion Detection: 12th International Symposium, RAID 2009, Saint-Malo, France, September 23-25, 2009. Proceedings 12*. Springer, 2009, pp. 21–40.

[40] F. Pendlebury, F. Pierazzi, R. Jordaney, J. Kinder, L. Cavallaro *et al.*, "Tesseract: Eliminating experimental bias in malware classification across space and time," in *Proceedings of the 28th USENIX Security Symposium*. USENIX Association, 2019, pp. 729–746.

[41] R. Coulter, J. Zhang, L. Pan, and Y. Xiang, "Domain adaptation for windows advanced persistent threat detection," *Computers & Security*, vol. 112, p. 102496, 2022.

[42] J. Khoury, "Jbeil source code," 2023. [Online]. Available: https://github.com/LMscope/Jbeil

[43] TrendMicro, "Cyberattack lateral movement explained," 2019. [Online]. Available: https://www.trendmicro.com/en_us/research/19/h/cyberattack-lateral-movement-explained.html

[44] CrowdStrike, "Lateral movement," 2022. [Online]. Available: https://www.crowdstrike.com/cybersecurity-101/lateral-movement/

[45] Mitre, "Initial access," 2019. [Online]. Available: https://attack.mitre.org/tactics/TA0001/

[46] Strongdm, "11 authentication-based vulnerabilities you need to know," 2022. [Online]. Available: https://www.strongdm.com/blog/authentication-vulnerabilities

[47] Mitre, "Credential access," 2019. [Online]. Available: https://attack.mitre.org/tactics/TA0006/

[48] ——, "Modify authentication process," 2022. [Online]. Available: https://attack.mitre.org/techniques/T1556/

[49] ——, "Brute force," 2022. [Online]. Available: https://attack.mitre.org/techniques/T1110/

[50] ——, "Forced authentication," 2020. [Online]. Available: https://attack.mitre.org/techniques/T1187/

[51] ——, "Multi-factor authentication interception," 2022. [Online]. Available: https://attack.mitre.org/techniques/T1111/

[52] ——, "Os credential dumping," 2022. [Online]. Available: https://attack.mitre.org/techniques/T1003/

[53] ——, "Account manipulation," 2022. [Online]. Available: https://attack.mitre.org/techniques/T1098/

[54] ——, "Use alternate authentication material: Pass the ticket," 2021. [Online]. Available: https://attack.mitre.org/techniques/T1550/003/

[55] H. Bian, T. Bai, M. A. Salahuddin, N. Limam, A. Abou Daya, and R. Boutaba, "Uncovering lateral movement using authentication logs," *IEEE Transactions on Network and Service Management*, vol. 18, no. 1, pp. 1049–1063, 2021.

[56] D. Liben-Nowell and J. Kleinberg, "The link-prediction problem for social networks," *Journal of the American Society for Information Science and Technology*, vol. 58, no. 7, pp. 1019–1031, 2007. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/asi.20591

[57] C. Bhagavatula, S. Feldman, R. Power, and W. Ammar, "Content-based citation recommendation," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 238–251. [Online]. Available: https://aclanthology.org/N18-1022

[58] S. M. Kazemi and D. Poole, "Simple embedding for link prediction in knowledge graphs," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31. Curran Associates, Inc., 2018. [Online]. Available: https://proceedings.neurips.cc/paper/2018/file/b2ab001909a8a6f04b51920306046ce5-Paper.pdf

[59] P. Xu, W. Hu, J. Wu, and B. Du, "Link prediction with signed latent factors in signed social networks," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser. KDD '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 1046–1054. [Online]. Available: https://doi.org/10.1145/3292500.3330850

[60] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations (ICLR)*, 2017.

[61] ——, "Variational graph auto-encoders," *NIPS Workshop on Bayesian Deep Learning*, 2016.

[62] M. Zhang and Y. Chen, "Link prediction based on graph neural networks," in *Advances in Neural Information Processing Systems*, 2018, pp. 5165–5175.

[63] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/5dd9db5e033da9c6fb5ba83c7a7ebea9-Paper.pdf

[64] A. Bojchevski and S. Günnemann, "Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking," in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. [Online]. Available: https://openreview.net/forum?id=r1ZdKJ-0W

[65] Y. Hao, X. Cao, Y. Fang, X. Xie, and S. Wang, "Inductive link prediction for nodes having only attribute information," in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, C. Bessiere, Ed. International Joint Conferences on Artificial Intelligence Organization, 7 2020, pp. 1209–1215, main track. [Online]. Available: https://doi.org/10.24963/ijcai.2020/168

[66] H. Bian, T. Bai, M. A. Salahuddin, N. Limam, A. Abou Daya, and R. Boutaba, "Technical report," 2020. [Online]. Available: http://bit.ly/tech_report_2020

[67] E. Rossi, B. Chamberlain, F. Frasca, D. Eynard, F. Monti, and M. Bronstein, "Temporal graph networks for deep learning on dynamic graphs," *arXiv preprint arXiv:2006.10637*, 2020.

[68] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[69] D. Y. Hancock, J. Fischer, J. M. Lowe, W. Snapp-Childs, M. Pierce, S. Marru, J. E. Coulter, M. Vaughn, B. Beck, N. Merchant *et al.*, "Jetstream2: Accelerating cloud computing via jetstream," in *Practice and Experience in Advanced Research Computing*, 2021, pp. 1–8.

[70] A. D. Kent, "Comprehensive, Multi-Source Cyber-Security Events," Los Alamos National Laboratory, 2015.

[71] grantho, "lateral-movement-simulator," 2023. [Online]. Available: https://github.com/grantho/lateral-movement-simulator

[72] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in neural information processing systems*, vol. 30, 2017.

[73] PyG, "Pyg is the ultimate library for graph neural networks," 2023. [Online]. Available: https://pyg.org

[74] ——, "models.graphsage," 2023. [Online]. Available: https://pytorch-geometric.readthedocs.io/en/latest/generated/torch_geometric.nn.models.GraphSAGE.html

[75] Google, "Pub/sub," 2023. [Online]. Available: https://cloud.google.com/pubsub

[76] Elastic, "Logstash," 2023. [Online]. Available: https://www.elastic.co/logstash/

[77] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 2016, pp. 855–864.

[78] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 701–710.

[79] A. Ahmed, N. Shervashidze, S. Narayanamurthy, V. Josifovski, and A. J. Smola, "Distributed large-scale natural graph factorization," in *Proceedings of the 22nd international conference on World Wide Web*, 2013, pp. 37–48.

Appendix A.
Meta-Review

A.1. Summary

The paper addresses the critical challenge of detecting lateral movement (LM) within enterprise networks. The authors propose a novel approach that uses dynamic temporal graph networks, inductive learning, and a threat sample augmentation approach to detect LM attacks from authentication logs. Their proposed system, `Jbeil`, demonstrates superior performance over traditional detection methods and promises to detect previously unseen attacks, marking a significant step forward in network defense.

A.2. Scientific Contributions

- Addresses a Long-Known Issue
- Provides a Valuable Step Forward in an Established Field
- Creates a New Tool to Enable Future Science

A.3. Reasons for Acceptance

1) The problem of detecting lateral movements in enterprise networks is a critical issue in cybersecurity. The paper offers a promising solution to this challenge.
2) The use of an inductive learning approach and the integration of dynamic temporal graph networks represents a significant methodological innovation.
3) The proposed system, `Jbeil`, exhibits improved detection accuracy compared to existing systems, particularly in scenarios involving unseen nodes.

A.4. Noteworthy Concerns

The paper doesn't discuss the system's efficiency and scalability, which are critical for practical deployments in large-scale networks. Providing information about the computational resources required and how well the system scales would provide a more comprehensive understanding of the tool's applicability.

Appendix B.
Response to the Meta-Review

Regarding the discussion of the system's efficiency and scalability, we have conducted additional experiments in which we examined `Jbeil`'s inference time according to various number of edges. We have revised Section 4.4.4 to showcase the efficiency and scalability of `Jbeil` while presenting insights on its inference time.