PIKACHU: Temporal Walk Based Dynamic Graph Embedding for Network Anomaly Detection

Ramesh Paudel
Department of Electrical and Computer Engineering
George Washington University
Washington, DC
rpaudel42@gwu.edu

H. Howie Huang
Department of Electrical and Computer Engineering
George Washington University
Washington, DC
howie@gwu.edu

NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium | 978-1-6654-0601-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/NOMS54207.2022.9789921

***Abstract*—Enterprise networks evolve constantly over time. In addition to the network topology, the order of information flow is crucial to detect cyber-threats in a constantly evolving network. Majority of the existing technique uses static snapshot to learn from dynamic network. However, using static snapshots is not sufficient as it largely ignores highly granular temporal information and leads to information loss due to approximation of aggregation granularity. In this work, we propose PIKACHU, a sophisticated, unsupervised, temporal walk-based dynamic network embedding technique that can capture both network topology as well as highly granular temporal information. PIKACHU learns the appropriate and meaningful representation by preserving the temporal order of nodes. This is important information to detect Advanced Persistent Threat (APT) as temporal order helps to understand the lateral movement of the attacker. Experiments on two open-source datasets: LANL and OpTC datasets demonstrated the effectiveness in detecting network anomalies. PIKACHU achieves True Positive Rate (TPR) of 95.1% in LANL and 98.7% on OpTC dataset. Furthermore, in the LANL dataset, it achieves a 4.65% reduction in False Positive Rate (FPR) despite similar area under ROC curve (AUC). In the OpTC dataset 16% improvement in AUC was obtained in comparison to the other state-of-the-art approaches.**

***Index Terms*—Network Anomaly, Graph Neural Network, Anomaly Detection**

I. INTRODUCTION

In real-time, the activities or events in computer networks are captured as a stream of logs. The interaction in such a network can be represented as a time-ordered sequence of edges forming a dynamically evolving network. For example, in an computer network, the process creation can be represented as a time-ordered sequence of edges where the edge represents the parent-child relationship. In this paper, we consider the anomaly detection problem in a dynamic graph where the goal is to detect whether the incoming edge is anomalous or not. Let us take an example of the Advanced Persistent Threat (APT) detection problem in a computer network. During APT campaigns, the attacker will initially authenticate the new system, get the foothold of the system, and traverse through the network gaining access to additional systems and credentials [1], [2]. The order of information flow is crucial to understand the lateral movement of the attacker. For example, during the day 1 attack campaign in the OpTC dataset (shown in Fig 1), the attacker initially connects to a host *sysclient0201* and download the PowerShell empire to modify the registry key and bypass user account control (UAC). The attacker then uses the privilege to harvest login credentials and use it to move laterally to *sysclient0402* and *sysclient0660* as an elevated agent. In the next stage, the attacker pivot to the domain controller *dc1.systemia.com* and finally spread to many other systems using the compromised user credentials. In a evolving network like above, besides the the network topology, the temporal order of event is important to accurately understand, model, and predict the overall network behavior [3], [4]. The assumption for APT detection is that the information flows and the lateral movement induced by malicious actors in the system are sufficiently different from the normal behavior of the system [5], [6].

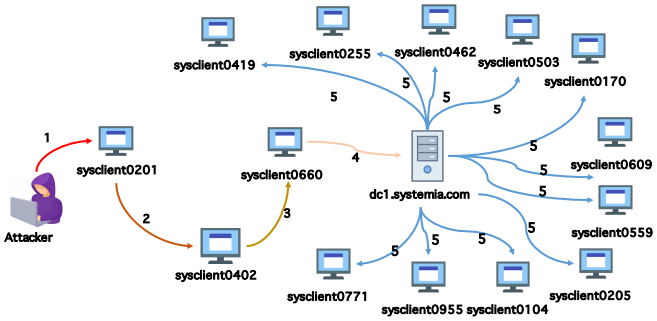


Fig. 1: Lateral movement in OpTC network during day 1 attack campaign. Initially, the attacker connects to *sysclient0201* (step 1) and uses privilege escalation to harvest login credentials. Using elevated privilege, attacker move to *sysclient0402* (step 2), *sysclient0660* (step 3), and the domain controller *dc1.systemia.com* (step 4) from where it spread to many other system (step 5).

Recently, various network representation technique has offered unprecedented possibilities to extract complex and meaningful patterns from the network [7], [8] and use them to identify malicious events. In dynamic network, heuristic rule-based [6], [9]–[14] and neural network-based [4], [5], [8], [15], [16] representation technique has been proposed for network intrusion detection. The problem with the heuristic-

based approach is that these approaches are rigid and focuses on specific types of anomalies. On the other hand, the neural network-based embedding technique faces two main challenges: (1) The embedding technique requires an elegant model to digest both topological and temporal information [8] and (2) Dividing the dynamic network as a sequence of a static snapshot of graphs leads to information loss due to approximation of aggregation granularity [4], [8], [17], [18]. In addition, selecting an appropriate aggregation granularity (e.g., minute, hour, day, or week) is itself a challenging problem that can lead to poor predictive performance or misleading results.

In this work, we present PIKACHU[1]: a unsupervised node embedding technique that capture both topological and temporal information. Initially, PIKACHU breaks down the graph stream into a sequence of static snapshots and uses the temporal walk to traverse each snapshot. Next, the Skip-gram model is used to capture the topological and granular temporal information by maximizing the likelihood of preserving the temporal order of the node. Finally, a gated-recurrent unit (GRU)-based auto-encoder is used for learning long-term temporal information. The architecture of PIKACHU is shown in Fig 2. Below are the contributions of this paper:

- We propose PIKACHU, a fully unsupervised dynamic node embedding technique for detecting cyber threats. As opposed to the previous dynamic embedding technique, PIKACHU learns node embedding by preserving the temporal order of the events to capture both topological and highly granular temporal information and can detect advanced cyber-threat with higher accuracy.
- We evaluate PIKACHU on LANL [19] and OpTC [20] datasets. Our method significantly outperforms all the baselines and demonstrates its capability of detecting anomalies efficiently than previous methods (4.65% reduction in False Positive Rate in LANL dataset and 16% improvement in AUC in OpTC dataset).

The remainder of this paper is organized as follows. Section 2 discusses the related work and section 3 describes the proposed methodology. We describe the dataset, experimental setup, and results in section 4 and we conclude in Section 5.

II. RELATED WORK

Our proposed framework is conceptually related to the anomaly detection problem in dynamic/streaming graphs. Techniques like GOutlier [9], CM-Sketch [10], SedanSpot [21], MIDAS [13], F-FADE [22], etc. has been proposed to detect anomalies in edge stream. GOutlier [9] uses reservoir sampling to maintain the structural summary of the graph stream and identifies unusual bridging edges as anomalies using the structural connectivity model. CM-Sketch [10] uses both the structural information and historical behavior to determine the anomalousness of an edge. SedanSpot [21] maintains a rate adjusted sample of edges which is then used to score the anomalousness of edges by giving diminishing importance to far-away neighbors. MIDAS [13] maintains the

[1]based on a Pokémon character from a movie Detective Pikachu

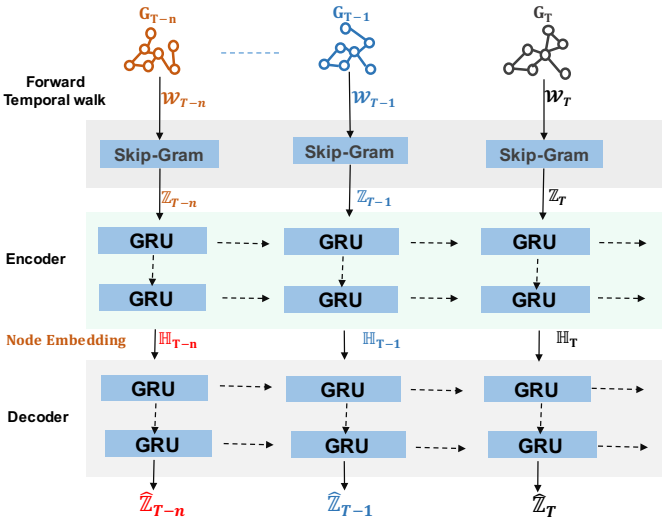


Fig. 2: Visual illustration of PIKACHU framework.

count-min-sketch of historical edge count as well as the current edge count and uses a hypothesis testing-based approach to detect anomalous micro-clusters (sudden burst of traffic). F-FADE [22] uses a novel frequency-factorization technique to efficiently model the time-evolving distributions of frequencies between node pairs. The anomalies are then determined based on the likelihood of the observed frequency of each incoming interaction. AnomRank [23] proposes an online algorithm for anomaly detection in a dynamic graph. StreamSpot [6], GODIT [12], and SnapSketch [14] uses hash-based sketching of shingles generated from random walks to convert graphs into sketch vector. SpotLight [11] composes a sketch containing total edge weights of directed query subgraphs chosen independently and uniformly at random. These methods mark entire snapshots as an anomaly by clustering sketch vectors. The above techniques are heuristic-based and are designed for detecting a specific type of anomalies.

Recent advances in graph embedding have inspired anomaly detection techniques based on network embedding. The anomaly detection approach based on network embedding has been proposed in [24]–[27] but they are designed for static graphs and ignore the dynamic and temporal nature of the network. Approaches like CAD [15] and Netwalk [5] proposes embedding-based anomaly detection technique on a dynamic graph. However, they cannot capture the short-term or long-term temporal patterns of nodes which are highly important in dynamic graph stream frameworks. Techniques like DynGem [28], EvolveGCN [29], tNodeEmbed [30], AddGraph [16], etc. divide graph streams into static snapshots and use a recurrent neural network to capture the temporal patterns. Kitsune [31] uses ensembles of auto-encoders for online anomaly detection. Zhou et al. [32] uses dynamic graph clustering with community detection model by ranking nodes according to their deviance from both their closest cluster centers plus historical behaviors to mark anomalies. LEADS [33] proposes graph embedding based on substructures and graph edit

distance for anomaly detection in heterogeneous graph stream. In addition, approaches like CTDNE [4], StreamWalk [17], TagGen [18], CAW [8], etc. propose using temporal walk to capture highly granular temporal interaction for accurate embeddings. CTDNE [4] uses the temporally valid walk to learn embedding representing graphs at the finest temporal granularity. StreamWalk [17] utilizes temporal walk to update the node embedding over time to reflect a change in network structure. TagGen [18] uses a bi-level self-attention mechanism trained using temporal random walks. CAW-N [8] propose an inductive learning model that uses causal anonymous walks (CAW) extracted from temporal random walks for modeling temporal networks.

III. Proposed Methodology

A. Background and Motivation

Initially, PIKACHU breaks down the graph stream into a sequence of static snapshots where each snapshot represents all edges that occur between a discrete time interval (e.g., minute, hour, or day). Each snapshot is traversed using *temporal walk*. A *temporal walk* traverses a graph in an increasing order of time. For example, a sequence of nodes $(v_{t_1}, v_{t_2}, ..., v_{t_n})$ in a graph $G_T = (V_T, E_T)$ is a valid temporal walk if $\{(v_{t_1}, v_{t_2}, t_{t_1}), (v_{t_2}, v_{t_3}, t_{t_2}), ..., (v_{t_n}, v_{t_{n+1}}, t_{t_n})\} \in E_T$ and $t_{t_1} \leq t_{t_2} \leq ... \leq t_{t_n}$. The sequence of nodes generated using temporal walk carries important information. In a computer network, an edge can represent an interaction between users and hosts and the temporal walk sequence can provide important information about the lateral movement of the user within the network. For instance, let us consider the sequence of events $e_1 = (u_1, h_1, 1)$, $e_2 = (h_1, h_2, 2)$, and $e_4 = (h_2, h_4, 4)$ from Fig 3. If the event e_1 corresponds to an user u_1 authenticating with host h_1 at time 1 and event e_2 and e_4 corresponds with host h_1 communicating with h_2 at time 2 and h_2 communicating with h_4 at time 4 respectively. Then the sequence $\{e_1, e_2, e_4\}$ could potentially be the lateral movement of user u_1 within the network after authenticating with host h_1. However, the event sequence $\{e_8, e_4\}$ such that $e_8 = (u_3, h_2, 8)$ and $e_4 = (h_2, h_4, 4)$ can be considered a valid walk by using a general random walk but does not provide information on valid temporal events. Embedding methods that ignore time order are highly likely to miss crucial information and are prone to learning inappropriate node embeddings that do not accurately capture the dynamics in the network [4], [17], [18]. PIKACHU tries to address this problem by introducing a temporal walk to traverse the graph within each snapshot.

B. Problem Setting

Let $\mathbb{G} = \{G_1, G_2, ..., G_T, ...\}$ be a graph stream where each G_T denotes a graph at discrete time interval T. We consider a graph $G_T = (V_T, E_T)$ as a generic directed graph where V_T and E_T are the set of nodes and edges respectively. An edge $e_t = (u, v, t) \in E_T$ means that the node u and node v have a connection at the timestamp t. The goal of this work is to learn a function $f : v \rightarrow \mathbb{R}^d$ that maps each node $v \in V_T$ to a $d-$dimensional time-dependent embedding using a temporal walks. The node embedding will then be used to detect anomalous edges in E_T.

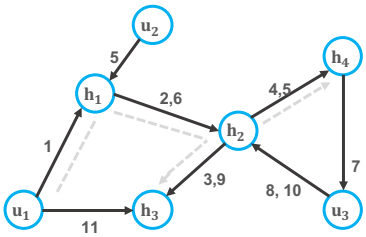


Fig. 3: Example of a dynamic graph. Edges are labelled by arrival time. The node sequence $(u_1 \rightarrow h_1 \rightarrow h_2 \rightarrow h_3)$ is a valid temporal walk. However, general random walk-based approach considers $(u_3 \rightarrow h_2 \rightarrow h_4)$ as a valid walk which is not true for the temporal walk.

Definition 1: (TEMPORAL WALK) A temporal walk from v_1 to v_k in G_T is a sequence of vertices $(v_1, v_2, ..., v_k)$ such that $(v_i, v_{i+1}) \in E_T$ for $1 \leq i < k$, and $T(v_i, v_{i+1}) \leq T(v_{i+1}, v_{i+2})$ for $1 \leq i < (k-1)$.

C. Graph Traversal using Temporal Walk:

For each snapshot G_T, we select the initial edge $e_t = (u, v, t) \in E_T$ using a uniform random distribution.

$$\mathcal{P}(e_t) = \frac{1}{|E_T|} \tag{1}$$

Next, to start the temporal walk from the initial edge e_t at time t, we select the temporal neighbors of outgoing node v at time t.

Definition 2: (TEMPORAL NEIGHBORHOOD) Temporal neighborhood of a node v at time t is denoted as $\tau_t(v)$ and defined as the set of nodes such that:

$$\tau_t(v) = \big\{(w', t') | e = (v, w', t') \in E_T \wedge t' > t\big\} \tag{2}$$

In Fig 3, let us suppose the initial edge be $(h_1, h_2, 2)$, then the temporal neighborhood $\tau_2(h_2) = \big\{(h_3, 3), (h_4, 4), (h_4, 5), (h_3, 9)\big\}$. It is to be noted that the same node can appear multiple times in $\tau_t(v)$ because there can be multiple activities between two nodes within a single snapshot T. For example, there can be multiple authentication attempts by a user on the host machine, or two hosts can exchange multiple packets of data. However, a generic random walk with no notion of time would only have $\tau_t(v) = \big\{h_3, h_4\big\}$. Thus, leading to information loss while a temporal walk can capture the temporal interaction in a highly granular manner. This can be really valuable information in solving complex problems like APT detection where the temporal walk can perfectly track the movement of the attacker from initial foothold to attacker's lateral movement through the network.

The next node in the temporal walk $\mathcal{W}_T = \{(w^1, t^1), ..., (w^\ell, t^\ell)\}$, where ℓ is the walk length, is chosen from $\tau_t(v)$ using a uniform random distribution.

$$\mathcal{P}(w) = \frac{1}{|\tau_t(v)|} \tag{3}$$

D. Skip-Gram for topological and short-term temporal dependency:

In natural language processing, the skip-gram model [34] is used to construct the vector representations of a word where the input is a text corpus and output is a set of vectors. The Skip-gram model maximizes the co-occurrence probability among the words that appear within a certain distance in a sentence. The idea is to use a skip-gram [34] model to capture network topology and short-term temporal features of the graph stream at time T. This technique has been successfully employed by graph embedding techniques like node2vec [35], Deepwalk [36], CTDNE [4], etc. in the past.

The learning task of a skip-gram model is to optimize the temporal order-preserving node embedding. We seek to maximizes the log-probability of observing a temporal walk $\mathcal{W}_T$ for a node v conditioned on its node embedding given by function $f : v \rightarrow \mathbb{R}^d$.

$$\max_f \sum_{v_j \in V_T} \log \mathcal{P}(\mathcal{W}_T | f(v_j)). \tag{4}$$

The optimization problem is made tractable by assuming the conditional independence of the node in the temporal walk $\mathcal{W}_T$ when observed with respect to the source node v.

$$\mathcal{P}(\mathcal{W}_T | f(v_j)) = \prod_{w^i \in \mathcal{W}_T} \mathcal{P}\big(w^i | f(v_j)\big) \tag{5}$$

The conditional likelihood of each source-neighborhood node pair can then be modeled using a softmax unit parametrized by a dot product of their embedding vectors.

$$\mathcal{P}(w^i | f(v_j)) = \frac{exp\big(f(w^i).f(v_j)\big)}{\sum_{v_k \in V_T} exp\big(f(v_k).f(v_j)\big)} \tag{6}$$

Finally, the optimization objective in Eq. 4 simplifies to:

$$\max_f \sum_{v_j \in V_T} \Big[-\log \mathcal{Z}_{v_j} + \sum_{w^i \in \mathcal{W}_T} f(w^i).f(v_j) \Big] \tag{7}$$

where $\mathcal{Z}_{v_j} = \sum_{v_k \in V_T} exp\big(f(v_j).f(v_k)\big)$ is a per-node partition function and can be approximated using negative sampling [37]. Skip-gram model will generate the node embedding $\mathbb{Z}_T$ at each snapshot T encoding spatial information and short-term temporal information. In a graph stream $\mathbb{G}$, $\mathbb{Z}_T$ represents state of nodes at a particular snapshot T.

In each graph G_T, if $\mathbb{S}$ be the space of all possible random walks and $\mathbb{S}_T$ be the space of all temporal walks then $\mathbb{S}_T \subseteq \mathbb{S}$. The temporal walk represents only a tiny fraction of possible random walk [4]. The probability of sampling the temporal walk using a generic random walk is extremely small given that generic random walk samples the set of random walk $\mathcal{S}$ from $\mathbb{S}$ [4]. Also, the majority of sampled walks $\mathcal{S}$ will represent the temporally invalid node sequence. Hence, using generic random walk for modeling dynamic graphs leads to inaccurate node embedding.

E. GRU for long-term temporal dependency:

The Skip-gram model focuses on modeling interactions among nodes in close proximity (within the same snapshot) to each other by preserving the temporal neighborhood. However, it cannot anticipate interactions that could occur beyond the current snapshot. During an APT campaign, after gaining control of the network, an attacker can remain ideal for hours, or sometimes even days or months to perform their mission. Therefore, it is crucial to understand the interaction over an extended period not just within the same snapshot. Recurrent Neural Network (RNN) like GRU [38] can be used to learn the long-term behavior patterns of the nodes. GRU is a variant of the RNN network which is simpler, more effective, and can record long-term information by avoiding gradient vanishing and exploding problems [38]. PIKACHU uses a GRU-based auto-encoder to encode long-term temporal features. The general layout of an auto-encoder consists of an input layer, an output layer, an encoder neural network, a decoder neural network, and a latent space. Let us suppose $\mathcal{E}$ is the encoder function and $\mathcal{D}$ decoder function. The encoder function $\mathcal{E}$ maps the static node embedding $\mathbb{Z}_T$ to the latent space $\mathbb{H}_T$ i.e. $\mathcal{E} : \mathbb{Z}_T \rightarrow \mathbb{H}_T$. Similarly, the decoder function $\mathcal{D}$ takes the latent space $\mathbb{H}_T$ and reconstructs the original data $\hat{\mathbb{Z}}_T$ i.e. $\mathcal{D} : \mathbb{H}_T \rightarrow \hat{\mathbb{Z}}_T$. The output is then compared with the initial data to compute the reconstruction error. The auto-encoder is trained by minimizing the reconstruction error as:

$$L(\mathbb{Z}_T, \hat{\mathbb{Z}}_T) = ||(\mathbb{Z}_T - \hat{\mathbb{Z}}_T)||^2 \tag{8}$$

F. Anomaly Detection

Akoglue et. al define anomaly as a rare instance that deviates significantly from normal behavior [39]. For any anomaly detection task, it is necessary to accurately model the normal behavior and then look for anomalous instances that deviate from the normal behavior. In graphs, anomaly detection on edge can be modeled using a conditional probability distribution of edges [25], in which the normal edges have a higher probability of occurrence whereas an edge with a lower probability is likely to be an anomaly.

Let us consider two vertices u and v. Let $N(u)$ represents the neighborhood of vertex u. The conditional probability distribution over all pairs of u, v in graph G_T denoted by $\mathcal{P}(v|u, N(u))_T$ represents the probability that there exists an edge between u and v at timestamp T. In other words, a probability distribution for each vertex is influenced by its neighbors because in graph structure data nodes in a densely connected group tend to have similar behaviors [25]. Furthermore, each edge (u, v) will appear in two probability distribution $\mathcal{P}(v|u, N(u))_T$ and $\mathcal{P}(u|v, N(v))_T$. Since $N(u)_T$ and $N(v)_T$ are highly likely to be different $\mathcal{P}(v|u, N(u))_T \neq \mathcal{P}(u|v, N(v))_T$. Finally, the anomalous edge can be detected using this distribution: for an edge (u, v), the smaller the

TABLE I: Summary of the dataset

	Total Events	# of Snapshot	Anomalous Events	Host	Anomalous Host
LANL	18.79 mil	711	702	2,184*	104*
OpTC	40.56 mil	110	21,641	621	23

*In LANL dataset, we use user-based authentication events. Therefore, the number corresponds to users instead of hosts.

$\mathcal{P}(v|u, N(u))_T$ and $\mathcal{P}(u|v, N(v))_T$, the more likely this edge is anomalous. A similar technique is used by Ouyang et al. [25] for detecting anomalous edges in static graphs.

In each snapshot, $\mathbb{H}_T$ maps each vertex $u \in V_T$ to a $d-$dimensional vector $\mathbb{H}_T(u) \in \mathbb{R}^d$. For each node u and its neighbors $N(u)_T$ we can get node embedding $\mathbb{H}_T(u)$ and $\mathbb{H}_T(w)|w \in N(u)_T$. Mean aggregate function g can be used to get the aggregated embedding representation $\mathcal{H}$.

$$\begin{aligned}\mathcal{H} &= g(\mathbb{H}_T(u), \mathbb{H}_T(w)|w \in N(u)_T) \\ &= \frac{1}{|N(u)_T| + 1}\left(\mathbb{H}_T(u) + \sum_{w \in N(u)_T} \mathbb{H}_T(w)\right)\end{aligned} \tag{9}$$

Aggregate node embedding $\mathcal{H}$ can then be mapped into the parameterized probability distribution $\hat{\mathcal{P}}(v|u, N(u))_T$ using a mapping function $\mathcal{F}$ such that:

$$\begin{aligned}f(v, u, N(u))_T &= \hat{\mathcal{P}}(v|u, N(u))_T \\ &= \mathcal{F}(\mathcal{H})|_v = Softmax(W.\mathcal{H})|_v\end{aligned} \tag{10}$$

where W is $d \times |V_T|$ trainable weight matrix that can be estimated from training data. The output of the softmax is the estimate of $\mathcal{P}(v|u, N(u))_T$. The weight matrix W can be estimated using the log-likelihood function such that:

$$L = \frac{1}{|V_T|} \sum_{(u,v) \in V_T} \ln \hat{\mathcal{P}}((v|u, N(u))_T; W) \tag{11}$$

This is basically a $|V_T|$-class classification problem where u and $N(u)_T$ are inputs and the objective is to predict which vertex have an edge connected with u. The loss function is

$$J = cross - entropy(f(u, N(u))_T, v) \tag{12}$$

The training sample consists of $(u, N(u)_T - v)$ as input and v is the target variable. Using all the neighbors in $N(u)_T$ can sometimes be computationally prohibitive. Therefore, we can sample s neighbors uniformly with replacement from the set of $N(u)_T - v$ as an approximation of $N(u)_T$. We refer s as a support set and the best value of s is estimated using parameter tuning during experimentation.

After training, the probability of the existence of an edge (u, v) can be estimated using the learned conditional probability estimation from the perspective of both u and v.

$$\begin{aligned}\mathrm{p}_v &= \hat{\mathcal{P}}(v|u, N(u))_T \\ \mathrm{p}_u &= \hat{\mathcal{P}}(u|v, N(v))_T\end{aligned} \tag{13}$$

The normal edge has a higher probability while anomalous edges have lower probabilities. Finally, the anomaly score for the edge is calculated using both p_v and p_u.

$$e_{score} = \frac{(1 - \mathrm{p}_v) + (1 - \mathrm{p}_u)}{2} \tag{14}$$

IV. Experimentation

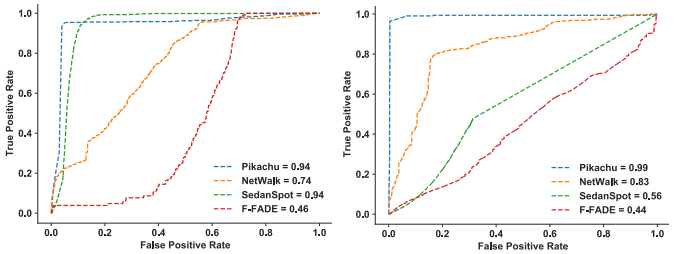
A. Dataset

LANL Dataset [19]: The subset of Comprehensive Multi-Source Cyber-Security Events dataset by Los Almos National Labs (LANL) is used. LANL data consists of de-identified event data across five data elements: window-based authentication events, process start and stop events, DNS lookups, network flows, and red-team activities over 58 days. In this experiment, we choose the window-based authentication events. We removed computer-account activities (user names ending with "$") and keep all human-driven activities. Also, we removed events from LOCAL, SYSTEM, ANONYMOUS, and ADMINISTRATOR accounts as they are not relevant for enterprise-level integration [40]. The red team activity has 749 events over 58 days that are marked as anomalous authentication. Out of 12.4K users, we selected all 104 anomalous users listed in red team activity and randomly selected 2,080 normal users (1 : 20 ratio) to avoid extremely unbalanced data. Our final data consist of 18.79 million authentication events from 2,184 users of which 702 events are anomalous.

OpTC Dataset [20]: DARPA Operationally Transparent Cyber (OpTC) dataset contains events from an enterprise network based on MITRE's Cyber Analytics Repository (CAR) data model. CAR model describe events in term of *object*, *action*, and *field*. *Object* is a visible entity like file, process, registry, task, thread, service, flow, etc. State change or event that happens on an object is an *action*. For example, file creation, file deletion, flow start, registry edit, etc. Observable properties of an object like flow start time, process id, registry type, etc. are *fields*. It consists of the data describing both benign and malicious behaviors. In this experiment, we use events related to flow start. i.e. FLOW object and START events. Here, source and destination IP are nodes and each flow start is an edge. The final data consist of 40.56 million flow events from 621 hosts of which 21.6K events (and 23 hosts) are anomalous (or compromised). The summary of the data used in the experiment is shown in Table I.

B. Experimental Setup

We divide both datasets into 1-hour snapshots for simulating dynamic graphs. For LANL, the first 40 hours (benign data before the first anomalous event) is used for training anomaly detectors and the rest is used for testing. Similarly, for the OpTC dataset, the first 3 days of benign activity are used for training, and the rest is used for testing. Anomalies are the events that are listed in red team activities. For LANL, red team activities are marked clearly. For OpTC, anomalous events correspond to the flow generated by the host and a specific process during or after the red team activity, i.e.,



(a) LANL Dataset (b) OpTC Dataset

Fig. 4: AUC curve using PIKACHU and baseline approaches

TABLE II: Anomaly detection results using PIKACHU and the baseline approaches. The result is the average of 5 run (± standard deviation).

Approach	TPR	FPR	AUC
LANL Dataset			
PIKACHU	**95.1%** (±0.11)	**5.05%** (±0.08)	**.94** (±0.004)
NetWalk	65.8% (±0)	34.2% (±0)	.74 (±0)
SedanSpot	90.3% (±0)	9.7% (±0)	.94 (±0)
F-FADE	43.6% (±0.85)	56.3% (±0.90)	.46 (±.012)
OpTC Dataset			
PIKACHU	**98.7%** (±0.01)	**0.42%** (±0.008)	**.99** (±.0003)
NetWalk	80.9% (±0.03)	19.1% (±0.06)	.83 (±.0002)
SedanSpot	47.8% (±0)	31.4% (±0)	.56 (±0)
F-FADE	46.6% (±0.70)	53.4% (±0.78)	.44 (±.014)

anomalous events are the events generated by the compromised host. It should be noted that both the graph embedding and anomaly detection approaches are unsupervised and the labels are required only to evaluate the performance of the proposed approach. We use a temporal walk of length (ℓ) 500. The autoencoder consists of 2 layer GRU (the first layer with 64 units and a second layer with 128 units) with a dropout layer (0.3 dropout rate) in between. The auto-encoder is trained using 50 epochs. For anomaly detection, the learning rate of 0.001 and 10 iteration is used to estimate the conditional probability distribution of edges. In addition, to find the influence of the size of the embedding dimensions d and the support set s, we evaluate the performance of PIKACHU over $d = \{50, 100, 150, 200, 300\}$ and $s = \{2, 5, 10, 15, 20, 25\}$. The optimal value of AUC is reported at $d = 100$ for both datasets. Similarly, the optimal value of AUC is reported at $s = 10$. Therefore, for final experimentation we use $d = 100$ and $s = 10$. Finally, we compare the performance against three state-of-the-art anomaly detection method including NetWalk [5], SedanSpot [21], and F-FADE [22].

C. Results

True Positive Rate (TPR), False Positive Rate (FPR), and area under the ROC curve (AUC) are used as our evaluation metrics. Table II summarizes the result of anomaly detection. PIKACHU has superior performance in terms of TPR, FPR, and AUC on both datasets against other baseline approaches. Fig 4 (a) shows the AUC plot obtained on the LANL dataset. The AUC plot on the OpTC dataset is shown in Fig 4 (b). As seen, PIKACHU have the best AUC. NetWalk performs fairly well with the second-best AUC on the OpTC as well as the LANL dataset. NetWalk initially generates the node embedding and dynamically updates the embeddings upon the change in a network by only updating the network walk affected by the new changes. It can learn the structural changes over time. However, it cannot learn the long-term temporal dependency. SedanSpot has similar AUC to that of PIKACHU on LANL dataset (see Table II). However, FPR is high on the LANL dataset and it performs poorly in the OpTC dataset. SedanSpot maintains the rate adjusted samples of edges and defines anomalies in terms of edge samples. A new edge is more anomalous if adding it to the sample produces a larger change in the distance between its incident vertices. It is primarily designed for identifying bridge edges and bursts of activities as anomalies. Therefore, it lacks generalization and is not adaptable to identify other types of anomalies. Also, their anomaly detection is based primarily on the structural changes in the graph. The performance of F-FADE is poor on both datasets. F-FADE uses the frequency-factorization technique to model the time-evolving distributions of frequencies of interactions between node-pairs where anomalies are determined based on the likelihood of the observed frequency of each incoming interaction. Furthermore, NetWalk uses reservoir sampling to maintain the compact record of vertex neighbors, SedanSpot keeps only the rate adjusted edge samples and F-FADE keeps only the most frequent interactions in memory. This use of sampling strategy by baseline approaches ultimately leads to information loss. However, we use all available edge samples to perform a temporal walk in each snapshot and generate the temporal order-preserving node embedding.

V. CONCLUSION

In this work, we propose PIKACHU, an unsupervised node embedding approach for anomaly detection in a dynamic graph that captures the short-term as well as long-term temporal dependencies. The learned embeddings are highly effective in learning the temporal order of edges that are vital for detecting network anomalies like lateral movement and APT. Experiments on real-world datasets demonstrated the effectiveness in detecting network anomalies. PIKACHU achieves 4.65% reduction in False Positive Rate in LANL dataset despite similar AUC and 16% improvement in AUC in OpTC dataset in comparison to the other state-of-the-art approaches. The results demonstrate that modeling granular temporal information in a dynamic graph is important for learning appropriate and meaningful network representations.

ACKNOWLEDGMENT

This work was supported in part by DARPA under agreement number N66001-18-C-4033 and National Science Foundation grants 1618706, 1717774, and 2127207. The views, opinions, and/or findings expressed in this material are those of the authors and should not be interpreted as representing the official views or policies of the Department of Defense, National Science Foundation, or the U.S. Government.

REFERENCES

[1] T. Yadav and A. M. Rao, "Technical aspects of cyber kill chain," in *International Symposium on Security in Computing and Communication*. Springer, 2015, pp. 438–452.

[2] 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.

[3] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world'networks," *nature*, vol. 393, no. 6684, pp. 440–442, 1998.

[4] G. H. Nguyen, J. B. Lee, R. A. Rossi, N. K. Ahmed, E. Koh, and S. Kim, "Dynamic network embeddings: From random walks to temporal random walks," in *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, 2018, pp. 1085–1092.

[5] W. Yu, W. Cheng, C. C. Aggarwal, K. Zhang, H. Chen, and W. Wang, "Netwalk: A flexible deep embedding approach for anomaly detection in dynamic networks," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 2672–2681.

[6] E. Manzoor, S. M. Milajerdi, and L. Akoglu, "Fast memory-efficient anomaly detection in streaming heterogeneous graphs," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 1035–1044.

[7] W. L. Hamilton, R. Ying, and J. Leskovec, "Representation learning on graphs: Methods and applications," *arXiv preprint arXiv:1709.05584*, 2017.

[8] Y. Wang, Y.-Y. Chang, Y. Liu, J. Leskovec, and P. Li, "Inductive representation learning in temporal networks via causal anonymous walks," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=KYPz4YsCPj

[9] C. C. Aggarwal, Y. Zhao, and S. Y. Philip, "Outlier detection in graph streams," in *2011 IEEE 27th international conference on data engineering*. IEEE, 2011, pp. 399–409.

[10] S. Ranshous, S. Harenberg, K. Sharma, and N. F. Samatova, "A scalable approach for outlier detection in edge streams using sketch-based approximations," in *Proceedings of the 2016 SIAM International Conference on Data Mining*. SIAM, 2016, pp. 189–197.

[11] D. Eswaran, C. Faloutsos, S. Guha, and N. Mishra, "Spotlight: Detecting anomalies in streaming graphs," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 1378–1386.

[12] R. Paudel, T. Muncy, and W. Eberle, "Detecting dos attack in smart home iot devices using a graph-based approach," in *2019 IEEE International Conference on Big Data (Big Data)*. IEEE, 2019, pp. 5249–5258.

[13] S. Bhatia, B. Hooi, M. Yoon, K. Shin, and C. Faloutsos, "Midas: Microcluster-based detector of anomalies in edge streams," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 3242–3249.

[14] R. Paudel and W. Eberle, "Snapsketch: Graph representation approach for intrusion detection in a streaming graph," in *Proceedings of the 16th International Workshop on Mining and Learning with Graphs (MLG)*, 2020.

[15] K. Sricharan and K. Das, "Localizing anomalous changes in time-evolving graphs," in *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, 2014, pp. 1347–1358.

[16] L. Zheng, Z. Li, J. Li, Z. Li, and J. Gao, "Addgraph: Anomaly detection in dynamic graph using attention-based temporal gcn." in *IJCAI*, 2019, pp. 4419–4425.

[17] F. Béres, D. M. Kelen, R. Pálovics, and A. A. Benczúr, "Node embeddings in dynamic graphs," *Applied Network Science*, vol. 4, no. 1, p. 64, 2019.

[18] D. Zhou, L. Zheng, J. Han, and J. He, "A data-driven graph generative model for temporal interaction networks," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 401–411.

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

[20] D. C. Weir, R. Arantes, H. Hannon, and M. Kulseng, "Operationally transparent cyber (optc)," 2021. [Online]. Available: https://dx.doi.org/10.21227/edq8-nk52

[21] D. Eswaran and C. Faloutsos, "Sedanspot: Detecting anomalies in edge streams," in *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2018, pp. 953–958.

[22] Y.-Y. Chang, P. Li, R. Sosic, M. Afifi, M. Schweighauser, and J. Leskovec, "F-fade: Frequency factorization for anomaly detection in edge streams," in *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, 2021, pp. 589–597.

[23] M. Yoon, B. Hooi, K. Shin, and C. Faloutsos, "Fast and accurate anomaly detection in dynamic graphs with a two-pronged approach," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 647–657.

[24] 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.

[25] L. Ouyang, Y. Zhang, and Y. Wang, "Unified graph embedding-based anomalous edge detection," in *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2020, pp. 1–8.

[26] F. Liu, Y. Wen, D. Zhang, X. Jiang, X. Xing, and D. Meng, "Log2vec: A heterogeneous graph embedding based approach for detecting cyber threats within enterprise," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 1777–1794.

[27] Z. Li, X. Cheng, L. Sun, J. Zhang, and B. Chen, "A hierarchical approach for advanced persistent threat detection with attention-based graph neural networks," *Security and Communication Networks*, vol. 2021, 2021.

[28] P. Goyal, N. Kamra, X. He, and Y. Liu, "Dyngem: Deep embedding method for dynamic graphs," *arXiv preprint arXiv:1805.11273*, 2018.

[29] A. Pareja, G. Domeniconi, J. Chen, T. Ma, T. Suzumura, H. Kanezashi, T. Kaler, T. B. Schardl, and C. E. Leiserson, "Evolvegcn: Evolving graph convolutional networks for dynamic graphs." in *AAAI*, 2020, pp. 5363–5370.

[30] U. Singer, I. Guy, and K. Radinsky, "Node embedding over temporal graphs," *arXiv preprint arXiv:1903.08889*, 2019.

[31] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: an ensemble of autoencoders for online network intrusion detection," *arXiv preprint arXiv:1802.09089*, 2018.

[32] R. Zhou, Q. Zhang, P. Zhang, L. Niu, and X. Lin, "Anomaly detection in dynamic attributed networks," *Neural Computing and Applications*, vol. 33, no. 6, pp. 2125–2136, 2021.

[33] S. Lagraa, K. Amrouche, H. Seba *et al.*, "A simple graph embedding for anomaly detection in a stream of heterogeneous labeled graphs," *Pattern Recognition*, vol. 112, p. 107746, 2021.

[34] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[35] 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*. ACM, 2016, pp. 855–864.

[36] 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*. ACM, 2014, pp. 701–710.

[37] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[38] 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.

[39] L. Akoglu, H. Tong, and D. Koutra, "Graph based anomaly detection and description: a survey," *Data mining and knowledge discovery*, vol. 29, no. 3, pp. 626–688, 2015.

[40] A. D. Kent, L. M. Liebrock, and J. C. Neil, "Authentication graphs: Analyzing user behavior within an enterprise network," *Computers & Security*, vol. 48, pp. 150–166, 2015.