PROV-GEM: Automated Provenance Analysis Framework using Graph Embeddings

Maya Kapoor, Joshua Melton, Michael Ridenhour, Siddharth Krishnan, Thomas Moyer
College of Computing and Informatics
University of North Carolina at Charlotte
Charlotte, NC, United States
{mkapoor1, jmelto30, mridenh7, siddharth.krishnan, tom.moyer}@uncc.edu

***Abstract*—Data provenance graphs, detailed traces of system behavior, are a popular construct to analyze and forecast malicious cyber activity like advanced persistent threats (APT). A critical limitation of existing analysis techniques is the lack of an automated analytic framework to predict APTs. In this work, we address that limitation by augmenting efficient capture and storage mechanisms to include automated analysis. Specifically, we propose PROV-GEM, a deep graph learning framework to identify malicious anomalous behavior from provenance data. Since data provenance graphs are complex datasets often expressed as heterogeneous attributed multiplex networks, we use a unified relation-aware embedding framework to capture the necessary contexts and associated interactions between the various entities manifest in the data. Furthermore, provenance graphs by nature are rich detailed structures that are heavily attributed compared to other complex systems that have been used traditionally in graph machine learning applications. Towards that end, our framework uniquely captures "multi-embeddings" that can represent varied contexts of nodes and their multi-faceted nature. We demonstrate the efficacy of our embeddings by applying PROV-GEM to two publicly available APT provenance graph datasets from StreamSpot and Unicorn. PROV-GEM achieves strong performance on both datasets with a 99% accuracy and 97% F1-score on the StreamSpot dataset, and a 97% accuracy and 89% F1-score on the Unicorn dataset, equaling or outperforming comparable state-of-the-art APT threat detection models. Unlike other frameworks, PROV-GEM utilizes an efficient graph convolutional approach coupled with relational self-attention to generate rich graph embeddings that capture the complex topology of data provenance graphs, providing an effective automated analytic framework for APT detection.**

***Index Terms*—cybersecurity, network embeddings, semantic attention, provenance graphs**

I. INTRODUCTION

Data provenance is the history of data as it is transformed by a system. Whole system provenance captures the activity of the entire system and this activity is represented as graphs detailing the causal relationships between the data and processes within the system. Processing these graphs using machine learning techniques has great potential for detecting advanced persistent threat (APT) attacks which easily evade more traditional detection techniques. The actors behind APT attacks are often supported by nation states or corporations with large amounts of computing, monetary, and technical or scientific resources. Targets of these entities can include other nation states, companies and their databases, critical infrastructure (such as power grids or water supply), embedded systems, and military technology. Thus, it is critical to have automated tools that can characterize provenance graphs as a rich set of features that can be used to develop downstream graph mining and machine learning applications —like attack detection.

In this work, we provide an end-to-end framework that processes provenance activity and is able to conduct predictive analytics on these graphs. A critical piece in our framework is PROV-GEM (**Prov**enance **G**raph **Em**bedding), which aggregates novel node-level multi-embeddings to capture provenance context. The detailed activity traces captured by system data collection tools render provenance graphs as heterogeneous, multiplex, and attributed complex systems. As such, these graphs cannot be easily captured as features for traditional machine learning approaches. Similar to real-world networks like social or biological networks, these provenance graphs are comprised of various entities that are characterized via different node types and edge types in the network of interest, and these properties need to be captured contextually in order to extract actionable insights from this data. Recent advances in graph neural networks have opened up exciting possibilities in feature representations that can be exploited for machine learning algorithms. PROV-GEM is a unified relation-aware contextual embedding that has been specifically designed for provenance graphs and the downstream application of attack detection.

Unlike more traditional cyber attacks, these attacks are often much more subtle, subverting common system activities. By tracking the entirety of a system's behavior, we are able to detect such attacks as they will deviate from the known-good behavior of the system. These deviations will be captured by the whole-system provenance capture mechanism, and manifest as additional nodes and edges within the graph. As an example, consider an attack where the adversary exploits a vulnerability in an application to trigger the running of another process. This anomalous behavior would be captured by the whole-system provenance capture mechanism and detected while analyzing the graph. However, manually inspecting

every such graph is impossible. Instead, what is needed are automated techniques to identify those actions which are not representative of nominal system behavior.

Oftentimes, the term *Advanced Persistent Threat* (APT) is used to classify the types of anomalous behavior that are difficult to detect using traditional detection mechanisms due to the nature of such attacks. These attacks leverage sophisticated attacks, such as zero-day exploits that have not been seen before (i.e., models cannot be trained on known attack data). Furthermore, they are *persistent*, meaning that these attacks are well-planned, often lasting months and even years. Adversaries willing to undergo such measures are often motivated by political and military goals, but the success of these attacks can also result in extreme economic impact, leakage of sensitive data, and even the collapse of critical infrastructure. The success of an APT relies on the actors' ability to remain undetected over time, so low-level or minor advances are made which are spaced out so as to not appear anomalous. Adversaries may even execute attacks which open back doors to systems which may be returned to later. Collectively, these patterns are known as *low and slow* attacks, which are difficult to detect in whole system provenance captures as data grows larger and larger over time and associations become more unreliable due to concept drift from normal system executions [1]. We detect this anomalous behavior by generating concise representations, or embeddings, of provenance graphs as input features to a neural classifier. Depending on the system and the data collection tools, provenance graphs may vary significantly in their size, so any automated threat detection system needs to be flexible and scalable. Additionally, data provenance graphs contain vast amounts of node and edge metadata, which is a rich source of information for automatic analytical frameworks that has been underutilized to date.

We demonstrate that our graph representation learning framework, PROV-GEM, is capable of encoding this rich information by combining multi-embeddings into tailored node representations through semantic self-attention, which are then further aggregated to produce a rich graph embedding. The graph embeddings generated by PROV-GEM are passed to a neural network classifier which is jointly trained with the graph encoder. The proposed framework contributes:

- The novel use of multi-embeddings to capture the complex topology of heterogeneous provenance graph data.
- The use of a smaller convolutional neighborhood reduces computational complexity and improves training stability.
- The emphasis on the multi-relational semantic context of nodes is effective for learning rich embeddings of provenance graphs.
- PROV-GEM is a generalizable and scalable framework that can be applied to any provenance graph dataset.

PROV-GEM demonstrates state-of-the-art performance on two APT datasets, achieving a 99% accuracy and 97% F1-score on the StreamSpot dataset, and a 97% accuracy and 89% F1-score on the Unicorn dataset.

II. Related Work

Earlier works in graph-based anomaly detection in APT attack scenarios used a K-means clustering approach to classify provenance graphs as malicious or benign. StreamSpot [2] was designed to address the stream-based nature of provenance data as it concerns APT detection using vector representations of graphs and locality-sensitive hashing (LSH) [3] on them. Frappuccino [4] introduces the CamFlow Linux security module for capturing data across all system interactions in their natural state and uses a dynamic sliding window to divide the exponentially large streaming graph data into manageable subsets where system behavior can be learned. Unicorn divides the graph into shards using a sliding windows algorithm for parallel processing [2]. They also run a comparison study of Unicorn to StreamSpot on StreamSpot's own public data set [1]. Unicorn was able to handle graphs with more diverse edge types and of a larger size than StreamSpot could process [5].

All previously mentioned works use a method of deriving subgraphs for processing. In contrast, our GNN-based approach is able to scale using techniques such as embedding nodes in parallel and running computations in batches so that the system can process and analyze the graph in its original format which better captures whole-system provenance. SIGL [6] uses a graph long-short term memory (LSTM) network for encoding and a standard multilayer perceptron for decoding which better captures the serial nature of provenance data over the preceding bag-of-subtrees approaches like StreamSpot and Frappuccino.

In this work, we approach the problem of APT attack detection by combining data provenance graphs with a deep graph convolutional network (GCN) approach designed for attributed heterogeneous multiplex networks. We formally describe the problem of APT attack detection as a graph classification task in graph representational learning. Our framework employs a GCN-based model that learns the various types of edges connecting nodes in the network and is capable of handling both the large size of networks and the heterogeneous nature of provenance graphs generated by data provenance collection tools.

III. Provenance Preliminaries

Data provenance is the history of data as it is transformed by a system. The history includes the data itself, the actions carried out, and the users who exerted control over those processes. Provenance pre-dates computing though, having roots in the fine art community, where tracking the lineage

[1] Streamspot Dataset: https://github.com/sbustreamspot/sbustreamspot-data

of works of art is critical to determining the authenticity of the work. Applying similar concepts of tracking the history of an object creates opportunities to protect and analyze data in new ways. Provenance is used in a myriad of applications including ensuring the integrity of systems and data [7], [8], system analysis and tuning [9], enabling reproducible scientific workflows [10]–[12], and forensic investigation of cybersecurity attacks [13], [14]. Each of these use cases requires analysis of the collected metadata. In this work, we focus specifically on detecting anomalous behavior within the system.

Provenance can be collected from a number of different sources, including the operating system [15], the network [16]–[18], and the applications that are processing data [19]. In order to ensure that tools can analyze this provenance data that varies in the granularity, there are specifications for representing provenance. The most commonly used provenance specifications are the Open Provenance Model (OPM) and the W3C-PROV family of specifications [20]. Both of these specifications model the provenance as a directed acyclic graph. Figure 1 shows an excerpt of provenance captured by the CamFlow system for the `wget` command line utility creating a UNIX domain socket (a file) on the system as it downloads a file from the internet (the file in this example is specified by the node labeled `[argv] argv0`).

IV. Problem Definition

We define the problem of anomalous behavior detection on provenance graphs as a graph classification task in graph representational learning. The complete set of notations used in this paper is given in Table I. A homogeneous network is defined as $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of all nodes in the graph, and $\mathcal{E}$ is the set of all edges in the graph. Based on this definition, we may define a data provenance graph as an attributed heterogeneous multiplex network.

Definition 1 (Attributed Network)**.** We define an attributed homogeneous network as $G = (\mathcal{V}, \mathcal{E}, \mathcal{A})$, where each node $v \in \mathcal{V}$ is associated with a set of node features $\mathcal{A} = \{\boldsymbol{x_i} \mid v_i \in \mathcal{V}\}$, where $\boldsymbol{x_i}$ is the feature vector associated with node v_i.

Definition 2 (Heterogeneous & Multiplex Network)**.** A heterogeneous network is defined as $G = (\mathcal{V}, \mathcal{E}, T_v, T_e)$. Each node $v \in \mathcal{V}$ is associated with mapping function $\phi(v) : \mathcal{V} \rightarrow T_v$, and each edge $e \in \mathcal{E}$ is associated with mapping function $\psi(e) : \mathcal{E} \rightarrow T_e$, where T_v and T_e denote the sets of node types and edge types, respectively.

Definition 3 (Relation Type)**.** Given a heterogeneous graph G with T_v node types and T_e edge types, we define a relation type r as a 3-tuple representing the source node type, edge type, and destination node type of the relation: $r = (t_v, t_e^{(v,u)}, t_u)$. The set R consists of all unique relation types present in the network, and R is defined by the data collection tool used for generating the provenance graph.

TABLE I
IMPORTANT NOTATIONS AND THEIR DEFINITIONS

Notation	Description
G	the input network
$\mathcal{V}, \mathcal{E}$	the node and edge sets of G
T_v, T_e	the node-type and edge-type sets of G
$\mathcal{A}$	the attribute set of G
R	the set of relation types of G
v, e	a node and edge in the graph
$\boldsymbol{x}$	the set of attributes of a node
$\mathcal{N}$	the neighborhood of a given node
k	neighborhood levels (or) hops
z	the latent representation of a graph
d	the dimension of the final overall embedding
d_a	the dimension of the attention vector

Problem Definition. Given a provenance graph as $G = (\mathcal{V}, \mathcal{E}, \mathcal{A}, R)$, where $\mathcal{V}$ is the set of nodes, $\mathcal{E}$ is the set of edges, $\mathcal{A}$ is the set of node attributes, and R is the set of all relation types, the goal of this work is to produce a rich, low-dimensional representation of G such that we can detect anomalous behavior in a system.

V. Datasets

We used data sets from the related works StreamSpot and Unicorn to train and evaluate our APT detection model. These datasets provide three attack scenarios in total along with benign system provenance data. All datasets contain a ratio of 5:1 benign graphs to attack graphs. In our pre-processing, we prune nodes and edges from the graph which are seen but do not have additional provenance data —for example those vertices which may appear as part of an edge but not be recorded separately as a vertex. The average size of these graphs per type is provided in Table II.

Drive-by-download Attack: From StreamSpot, we combine the normal web activity browsing Gmail, YouTube, and the CNN website as well as video gaming and downloading software to create a benign group. The attack scenario is a drive-by-download attack that gained root access to the host machine by exploiting an Adobe Flash vulnerability created by visiting a malicious URL. The graphs are constructed from StreamSpot's summarized publication of their original dataset, which contain the source node ID, destination node ID, source node provenance type, destination node provenance type, edge provenance type, and an ID indicating which graph the pairing belonged to. We develop a parsing script to transform the data into directed NetworkX [21] graphs and one-hot encode the node and edge types as feature vectors.

Trojan Horse Attack: For the Unicorn dataset, we analyzed both simulated advanced persistent threat (APT) supply chain attack scenarios [22]. The first scenario is a trojan horse exploit. A continuous integration server uses wget version 1.17 to retrieve packages from repositories. The attacker knows that this version is vulnerable to remote file upload if a malicious

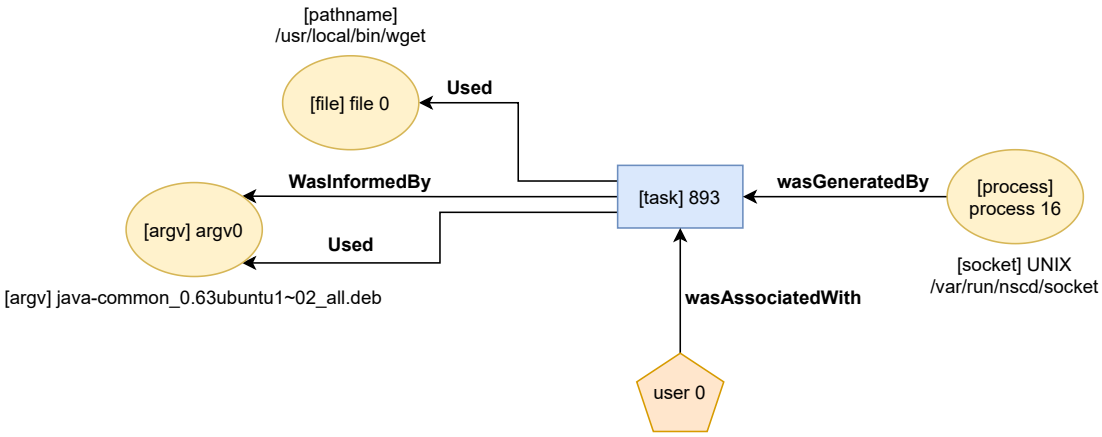

Fig. 1. Example provenance graph showing the `wget` process creating a UNIX domain socket.

TABLE II
SUMMARIZATION OF GRAPHS

Dataset	Class Type	Scenarios	# of Graphs	Avg. # Nodes	Avg. # Edges
StreamSpot	Benign	YouTube, GMail, Video Game, Download, CNN	500	8,315	173,857
	Attack	Drive-by-download Attack	100	8,891	28,424
Unicorn	Benign	Download (wget)	125	265,424	975,226
	Attack	Trojan Horse Attack	25	257,156	957,968
		Shellshock Attack	25	243,658	949,887

URL to a compromised server is requested. Thus, the attacker plants a remote access trojan (RAT) into one of the software packages and reconfigures the repository so that it directs the downloader to the attacker's FTP server and package holding the RAT. A reverse TCP shell session is created on the CI server, which allows the attacker to modify the server configuration and control deployment output [5].

Shellshock Attack: In the second scenario, a GNU Bash (v4.3) vulnerability is exploited to allow remote arbitrary code execution to compromise the supply chain. The benign scenarios involved downloading software from a non-compromised, trusted source.

The original provenance data are captured in W3C-PROV JSON [20] format using the CamFlow (v5.0) [23] Linux Security Module for whole-system data provenance collection. From this data, we construct a bidirectional graph which can be contextually analyzed. This completeness is ideal for APT attack detection because it represents the long-term, cause-and-effect relationships of processes over time [24] and eliminates race conditions by working in the kernel space rather than the system call interface [25].

In order to convert the CamFlow data into a graph for representation learning, we develop a parsing tool to derive source and destination nodes, edges, and their provenance types. We model the data using the W3C provenance labels for nodes and relations [20]. W3C PROV defines three node types: `agent`, `entity`, and `activity`. Additionally, there are relations such as `wasDerivedFrom`, `used`, `wasGeneratedBy`, and more defined in W3C PROV documentation [20]. The CamFlow LSM records node and edge types of specific system components and their processes which are well-defined in the documentation [23]. Examples include `process_memory`, `file`, `argv`, and `link` for nodes and `proc_read`, `exec`, and `clone_mem` for edges. We created NetworkX [21] graphs for the 125 benign scenarios and 25 for each attack type. CamFlow provides additional provenance data such as jiffies counts, security contexts, modes, access and modification times, dates, inode information, and system-specific IDs which may be used for more insight into system behavior and as higher dimensions for learning [23].

VI. METHODOLOGY

In this section, we outline the architecture and training procedure of the proposed GNN-based framework for detecting system attacks. In provenance graphs, nodes are involved in numerous relation types and can be related to each other in multiple ways. For example, Figure 1 is a subgraph from our training data which shows multiple relations between entities. Additionally, temporal properties of the graph are key in understanding system execution and influence nodes have on one another.

The system features recorded by CamFlow varies depending on provenance data type; therefore, we use locality-sensitive hashing (LSH) in order to normalize the data into buckets which can be used as an additional layer in the learning

model. The use of similarity hashing is logical here for several reasons. First, for temporal provenance data values like jiffies and access/modification time, it is intuitive that events which occur more closely within the same time span should be more likely associated. Data or events which share resources like information nodes are also likely related. Resources which are set with the same security contexts and access rights should also be acknowledged. Similarity hashing thus provides a method of converting the varied length, high-dimensional vectors generated by CamFlow into a fixed-length, lower-dimensional space while preserving like features.

We used TLSH [26], an algorithm developed by Trend Micro, to create similarity digests of 70 bytes in length. In this process, a sliding window of size 5 is used to populate an array of bucket counts. The quartiles of this array are then calculated. Next, the 3 byte digest header is derived from a checksum of the byte string, the logarithm of the byte string length, and two 16 bit quantities derived from the quartiles. The remainder of the digest is calculated via logic derived from the bucket array and appended with the header to form the final digest. The two Unicorn datasets created an average of 9 unique similarity digests for node features per graph and an average of 7 unique similarity digests for edge features. We used these hashes to group nodes and edges into buckets and encoded this representation into a feature vector for additional dimensionality in the graph framework.

The proposed model encodes this complex data into low-dimensional spatial representations of graphs by learning an inductive transformation function on a graph G. Using the message passing paradigm described by [27], our model generates a set of relation-specific latent representations of for each node based on its local neighborhood on each relation-type respectively. We create a single optimal representation for each node by using a semantic attention mechanism to combine the aforementioned sets of node embeddings. To produce a representation for the entire graph, we pass the set of optimal node embeddings to a permutation-invariant graph readout function which produces an aggregated, low-dimensional embedding of the entire graph.

The initial representation $\boldsymbol{h}_v^0$ for each node v is given by the node feature vector $\boldsymbol{x}_v$. We utilize the message passing function proposed by [28], which has been shown to be theoretically as expressive as the Weisfeiler-Lehman graph isomorphism test and the WL subtree kernel. The neighbor message function is defined as:

$$\phi_r^k(\mathcal{N}(v,r)) = \mathrm{MLP}^k\left(\boldsymbol{h}_v^{(k-1)} + \sum_{u \in \mathcal{N}(v,r)} \boldsymbol{h}_u^{(k-1)}\right) \quad (1)$$

where $\mathcal{N}(v,r)$ is node v's local neighborhood on relation r, $\boldsymbol{h}_v^{(k-1)}$ is the self-node v's representation from the previous layer, $\boldsymbol{h}_u^{(k-1)}$ is each neighbor node u's latent representation from the previous layer. We utilize a two layer MLP with an ELU [29] nonlinear activation function.

Each node in a provenance graph participates in multiple semantic relationships defined across the various relations in the network. The above message function reflects only a single aspect of the node's semantic context in the network. To learn a more comprehensive embedding for a node, we apply semantic attention [30] to the sequence of relation-specific latent representations of a node. In provenance graphs, nodes may be similar to one another in different ways, depending on the multiple relations in which each node is involved. For example, the same file *F1* may be generated by a task *T* which is read by process *P*. Process *P* may write another file *F2*. In provenance capture data, nodes and edges will have their own system-specific and type-specific attributes which do not map cleanly to one another without engineering and design. Due to the system privileges of *P*, *F1* and *F2* may share similar security contexts but could be completely unrelated otherwise. *T* may modify *F1* at a later time or modify its privileges, and thus the relations intertwine and grow more complicated over time. Our framework explicitly models the complex position of each node within this heterogeneous network by applying relation-specific message function in combination with relational self-attention, which blends the various relational contexts while learning the importance of each specific relation to each node.

First, we stack each of the relation-specific representations of node v from equation 1 as the sequence:

$$\tilde{\boldsymbol{h}}_v^k = \mathrm{CONCAT}(\phi_r^k(\mathcal{N}(v,r)) \mid \forall\, r \in R) \quad (2)$$

To learn the optimal set of attention weights for each relation, we transform the embeddings using a nonlinear transformation and compute the similarity with a relation-level attention vector. The attention weights are obtained by normalizing the above similarity computation using the softmax function:

$$\boldsymbol{a}_v^k = \mathrm{softmax}(\mathbf{w}^k \cdot \tanh(\boldsymbol{W}^k \cdot \tilde{\boldsymbol{h}}_v^k)) \quad (3)$$

where $\mathbf{w}^k$ is a trainable relation attention matrix of shape $|R| \times d_a$ and $\boldsymbol{W}^k$ is a trainable transformation matrix with size $|R| \times d \times d_a$. The final embedding for node v is the linear combination of the learned relation attention weights with the relation-specific latent representations:

$$\boldsymbol{h}_v^k = \boldsymbol{a}_v^k \cdot \tilde{\boldsymbol{h}}_v^k \quad (4)$$

To produce a feature representation of the graph, a readout function is applied to aggregate the individual node representations for each node in the network:

$$\boldsymbol{z}_{\mathcal{G}} = \sum_{v \in \mathcal{G}} \boldsymbol{h}_v^k \quad (5)$$

A. Model Optimization and Classification Task

To address the problem of threat detection using provenance graphs, we utilize the proposed embedding framework to generate a low-dimensional representation of each graph. These graph embeddings are passed through (1) a two layer MLP with a ReLU activation between layers and (2) a final sigmoid activation after the last layer to predict the probability of the positive class. We use binary cross entropy loss to jointly train the graph encoder and neural network classifier.

An additional advantage of employing a GNN-based approach is the scalability and parallelizability of such approaches for handling large heterogeneous networks. Where other threat detection frameworks on provenance graphs [2], [4]–[6] have focused on graph summarization and other techniques to reduce the size of inputs to their classification models, GNN models can easily batch nodes using message flow graphs and compute the individual node embeddings for large networks in parallel. This allows our framework to classify provenance graphs in their original heterogeneous format, incorporating the full richness of information captured by the system traces used to produce the provenance graphs.

VII. RESULTS

We evaluate our framework on the two publicly available datasets described in Section V. We pose the problem of provenance graph threat detection as a binary graph classification task, where the positive class is graphs derived from an attack and the negative class are benign data provenance graphs. In our experimental setup, we utilize 5-fold cross validation with both the StreamSpot and supply chain APT datasets, such that our model is trained on 80% of the available provenance graphs and 20% of the graphs are held out as an unseen test set. We obtain the classification accuracy, precision, recall, and F1 scores for each test fold and report the average of each metric.

Table III illustrates the performance of our framework on attack detection on the StreamSpot dataset. We compare against the baseline results reported by StreamSpot's own threat detection model as well as against a state-of-the-art detection model that also uses a provenance graph-based approach [5]. The isolated scenarios present in this dataset may not be representative of typical workloads in modern systems, but they provide a standard for comparing the relative performance of different systems and may be interpreted as a similar design pattern for today's microservice architectures [31]. As demonstrated in Table III, our model achieves near perfect performance on the graph classification task. In addition, our model requires only a single hop local neighborhood when applying the graph convolutional filters, clearly indicating the benefits of representing data provenance graphs as heterogeneous multiplex networks and modeling the semantic context of nodes over the various relation types captured by system data collection tools.

TABLE III
EXPERIMENTAL RESULTS ON ATTACK CLASSIFICATION ON THE STREAMSPOT DATASET. BASELINE ACCURACY AND PRECISION FOR STREAMSPOT ESTIMATED BY [5]. [32] DID NOT REPORT RECALL NOR F-SCORE.

Model	Accuracy	Precision	Recall	F1-Score
StreamSpot (baseline)	0.66	0.74	N/A	N/A
Unicorn (k = 1)	0.51	1.0	0.60	0.68
Unicorn (k = 3)	0.98	0.93	**0.96**	0.94
PROV-GEM (k = 1)	**0.99**	**1.0**	0.94	**0.97**

TABLE IV
EXPERIMENTAL RESULTS ON ATTACK CLASSIFICATION ON THE SUPPLY-CHAIN APT ATTACK SCENARIO.

Model	Accuracy	Precision	Recall	F1-Score
Unicorn (k = 3)	0.90	0.85	**0.96**	**0.90**
PROV-GEM (k = 1)	**0.97**	**1.0**	0.80	**0.89**

The experimental results on supply chain attack dataset are presented in Table IV. This dataset contains only 150 total graphs, which is a very limited dataset size for training a deep neural network based model. The small size of the total dataset coupled with the heterogeneity and large individual size of the provenance graphs could lead to training instability and significant variance in model performance. Despite these challenges, our model remains competitive with state-of-the-art threat detection models again while considering only the immediate 1-hop neighborhood of vertices as compared to the 3-hop neighborhood required by the Unicorn model.

The WL subtree kernel algorithm used by Unicorn belongs to a class of methods based on the Weisfeiler-Lehman isomorphism test [33] and well known for their discriminative power. Such algorithms rely on preprocessing of the graph to capture the local structure surrounding each vertex, as in [5]. Graph neural network (GNN) approaches are similarly characterized by their ability to represent the local topology surrounding vertices through the use of trainable convolutional filters that operate on each node's local neighborhood in the network. Such approaches have been successfully applied to many downstream graph learning tasks [34], including graph classification, and recent advances in GNN models have developed algorithms that are provable as expressive as the WL isomorphism test and related WL kernel methods [28].

We employ such a GNN model for our threat detection model, and as is evident from Tables III and IV, our framework is able to equal or outperform state-of-the-art WL-based models. In addition, PROV-GEM accomplishes this high level of performance with only the local 1-hop node neighborhood as compared with the 3-hop neighborhood required for the WL kernel algorithm, implying that our heterogeneous GCN-based approach is more efficient and better able to capture the multi-faceted context of nodes within provenance graphs. Rather

than focusing on the breadth of a node's context in the network and increasing computational and memory complexity through the use of multiple k-hop neighborhoods for each node, we instead emphasize the semantic context of nodes in provenance graphs by analyzing the multi-relational heterogeneous context of nodes through relation-specific convolutional filters and a semantic self-attention mechanism.

As our results on the StreamSpot and supply chain APT attack datasets indicate, such heterogeneous GCN models may be better suited for APT attack detection with data provenance graphs than WL kernel based methods as GCN models are trivially parallelizable through node batching and message flow graphs and can easily scale to handle large heterogeneous networks while maintaining the generalizability of WL-based methods for graph classification tasks [27], [28], [35].

VIII. Conclusions & Future Work

Graph neural networks are an apt tool for anomaly detection problems because system processes represented as multiplex, heterogeneous, attributed graphs depict characteristics and routines which can be studied through machine learning techniques. Anomalous behavior creates sub-portions of the graph and inconsistent features which are detected by the GNN framework. PROV-GEM is especially well-suited for this problem because of its parallelization and scalability which allow us to consider the entire graph instead of taking a sub-graph approach which could prevent detection of anomalous behavior between windows [2]. This system also uses a smaller convolution window which optimizes the training process compared to previous solutions. Furthermore, the ability to train on benign data allows PROV-GEM to detect anomalies which have not been seen before, thus potentially catching novel or zero-day attacks that are possible in APT scenarios, providing additional evidence of the generalizability of this framework.

PROV-GEM also contributes to current research in graph representational learning by using multi-embeddings to capture heterogeneous provenance graph data, as well as the use of locality-sensitive hashing to reduce varied attribute data to a fixed-length similarity context. We intend to explore other methods of fixed-length node feature extraction, including spectral and structure-based methods. At present, PROV-GEM employs standard supervised training using cross entropy to jointly train the graph encoder and neural network classifier. In future work, development of graph reconstruction loss functions for data provenance graphs will allow our framework to utilize self-supervised training and to expand the amount of training data available for model optimization beyond the limited labeled APT datasets currently publicly available. In addition, recent advances in attention mechanisms promise to further reduce the computational complexity of our framework by introducing more efficient means of computing tailored node-specific embeddings across the multi-relational semantic contexts present in provenance graphs.

References

[1] A. Tsymbal, "The problem of concept drift: definitions and related work," 2004.

[2] E. A. Manzoor, S. Momeni, V. N. Venkatakrishnan, and L. Akoglu, "Fast memory-efficient anomaly detection in streaming heterogeneous graphs," 2016.

[3] P. Indyk and R. Motwani, "Approximate nearest neighbors: Towards removing the curse of dimensionality," in *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, ser. STOC '98. New York, NY, USA: Association for Computing Machinery, 1998, p. 604–613. [Online]. Available: https://doi.org/10.1145/276698.276876

[4] X. Han, T. Pasquier, T. Ranjan, M. Goldstein, and M. Seltzer, "Frappuccino: Fault-detection through runtime analysis of provenance," in *9th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 17)*. Santa Clara, CA: USENIX Association, Jul. 2017. [Online]. Available: https://www.usenix.org/conference/hotcloud17/program/presentation/han

[5] X. Han, T. F. J. Pasquier, A. Bates, J. Mickens, and M. I. Seltzer, "UNICORN: runtime provenance-based detector for advanced persistent threats," *CoRR*, vol. abs/2001.01525, 2020. [Online]. Available: http://arxiv.org/abs/2001.01525

[6] X. Han, X. Yu, T. Pasquier, D. Li, J. Rhee, J. Mickens, M. Seltzer, and H. Chen, "Sigl: Securing software installations through deep graph learning," 2021.

[7] M. Allen, A. Chapman, L. Seligman, and B. Blaustein, "Provenance for collaboration: Detecting suspicious behaviors and assessing trust in information," in *Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2011 7th International Conference On*, Oct. 2011, pp. 342–351.

[8] A. Martin, J. Lyle, and C. Namilkuo, "Provenance as a security control," in *Proceedings of the 4th USENIX Conference on Theory and Practice of Provenance*, ser. TaPP'12. Berkeley, CA, USA: USENIX Association, 2012, pp. 3–3.

[9] P. Missier, "Incremental workflow improvement through analysis of its data provenance," 2011.

[10] D. Crawl, J. Wang, and I. Altintas, "Provenance for MapReduce-based data-intensive workflows," in *Proceedings of the 6th Workshop on Workflows in Support of Large-Scale Science*, ser. WORKS '11. New York, NY, USA: ACM, 2011, pp. 21–30.

[11] R. Ikeda, H. Park, and J. Widom, "Provenance for generalized map and reduce workflows," in *In Proc. Conference on Innovative Data Systems Research (CIDR*, 2011.

[12] P. J. Guo and M. Seltzer, "Burrito: Wrapping your lab notebook in computational infrastructure," in *Proceedings of the USENIX Workshop on the Theory and Practice of Provenance (TaPP)*, ser. USENIX Workshop on the Theory and Practice of Provenance (TaPP), Jun. 2012.

[13] K. H. Lee, X. Zhang, and D. Xu, "High accuracy attack provenance via binary-based execution partition." in *NDSS*. The Internet Society, 2013.

[14] S. Ma, X. Zhang, and D. Xu, "ProTracer: Towards practical provenance tracing by alternating between logging and tainting," in *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2016.

[15] D. J. Pohly, S. McLaughlin, P. McDaniel, and K. Butler, "Hi-Fi: Collecting high-fidelity whole-system provenance," in *Proceedings of the 28th Annual Computer Security Applications Conference*, ser. ACSAC '12. New York, NY, USA: ACM, 2012, pp. 259–268.

[16] A. Bates, K. Butler, A. Haeberlen, M. Sherr, and W. Zhou, "Let SDN be your eyes: Secure forensics in data center networks," in *Proceedings of the NDSS Workshop on Security of Emerging Network Technologies (SENT'14)*, San Diego, CA, Feb. 2014.

[17] A. Chen, Y. Wu, A. Haeberlen, W. Zhou, and B. T. Loo, "Differential provenance: Better network diagnostics with reference events," in *Proceedings of the 14th ACM Workshop on Hot Topics in Networks*, ser. HotNets-XIV. New York, NY, USA: ACM, 2015, pp. 25:1–25:7.

[18] W. Zhou, Q. Fei, A. Narayan, A. Haeberlen, B. T. Loo, and M. Sherr, "Secure network provenance," in *Proceedings of the 23rd ACM Symposium on Operating Systems Principles (SOSP)*, Cascais, Portugal, Oct. 2011.

[19] D. Tariq, M. Ali, and A. Gehani, "Towards automated collection of application-level data provenance," in *Proceedings of the 4th USENIX Conference on Theory and Practice of Provenance*, ser. TaPP'12. Berkeley, CA, USA: USENIX Association, 2012, pp. 16–16.

[20] W3C-PROV Working Group, "PROV-Overview: An Overview of the PROV Family of Documents," https://www.w3.org/TR/prov-overview/, 4 2013.

[21] NetworkX developer team, "Networkx," 2014. [Online]. Available: https://networkx.github.io/

[22] X. Han, "Wget Dataset," 2018. [Online]. Available: https://doi.org/10.7910/DVN/IA8UOS

[23] T. F. J. Pasquier, X. Han, M. Goldstein, T. Moyer, D. M. Eyers, M. I. Seltzer, and J. Bacon, "Practical whole-system provenance capture," *CoRR*, vol. abs/1711.05296, 2017. [Online]. Available: http://arxiv.org/abs/1711.05296

[24] T. Pasquier, X. Han, T. Moyer, A. Bates, O. Hermant, D. Eyers, J. Bacon, and M. Seltzer, "Runtime analysis of whole-system provenance," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 1601–1616. [Online]. Available: https://doi.org/10.1145/3243734.3243776

[25] T. Jaeger, A. Edwards, and X. Zhang, "Consistency analysis of authorization hook placement in the linux security modules framework," *ACM Trans. Inf. Syst. Secur.*, vol. 7, no. 2, p. 175–205, 2004. [Online]. Available: https://doi.org/10.1145/996943.996944

[26] J. Oliver, C. Cheng, and Y. Chen, "Tlsh – a locality sensitive hash," in *2013 Fourth Cybercrime and Trustworthy Computing Workshop*, 2013, pp. 7–13.

[27] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proceedings of the 34th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, D. Precup and Y. W. Teh, Eds., vol. 70. PMLR, 06–11 Aug 2017, pp. 1263–1272.

[28] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *ICLR*, 2019.

[29] D. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2016.

[30] Z. Lin, M. Feng, C. D. Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio, "A structured self-attentive sentence embedding," *ArXiv*, vol. abs/1703.03130, 2017.

[31] D. Namiot and M. sneps sneppe, "On micro-services architecture," *International Journal of Open Information Technologies*, vol. 2, pp. 24–27, 09 2014.

[32] X. Han, "StreamSpot Dataset," 2018. [Online]. Available: https://doi.org/10.7910/DVN/83KYJY

[33] B. Weisfeiler and A. Leman, "The reduction of a graph to canonical form and the algebra which appears therein," *NTI, Series*, vol. 2, no. 9, pp. 12–16, 1968.

[34] Z. Zhang, P. Cui, and W. Zhu, "Deep learning on graphs: A survey," *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–1, 2020.

[35] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 974–983.