Few-Shot Malware Variant Detection via Heterogeneous Graph Contrastive Learning

***Abstract*—Malware variant attacks are still a major threat in today's Internet ecosystem. However, prior arts on malware variant detection has thus far concentrated on supervised learning to identify the malware variants over large amount of labeled samples and very limited work can be transferred to detect few-shot advance variants (e.g., zero-day attack or stealthy malware), which are ubiquitous in the real-world attack scenarios and extremely scarce the sufficient samples and ground-truth labels. In this paper, we propose a self-supervised Heterogeneous Graph Contrastive Learning framework for advanced Malware variant detection (MalHGCL) with various types of malware entities and relationships, which belongs to a instance-instance discrimination. MalHGCL first designs two levels of heterogeneous graph data augmentations, such as API attribute masking, interaction enhancing, and meth-path sampling, to generate elaborate positive and negative pairs for each family instance, incorporating semantic prior and structural prior in the heterogeneous graph, respectively. Then MalHGCL utilizes heterogeneous graph contrastive learning to empower graph isomorphism networks (GINs) to learn the matching and mismatching graph-level representations between positive instance pairs and negative instance pairs in a self-supervised way. Finally, the experimental results show that the proposed MalHGCL on diverse datasets can achieve competitive or superior advanced variant detection performance than the state-of-the-art supervised detection methods and the generative detection model.**

***Index Terms*—malware variant detection, few-shot learning, self-supervised learning, heterogeneous graph augmentation, graph contrastive learning.**

I. Introduction

The long-lasting race on security warfare has faced new cliffs since the attacker community is increasingly focusing on sophisticated attack vectors to create the advanced variants [1], [2], [3]. Recent report [4] has estimated that over 350,000 new malware are registered daily, in which more than half are variants from the existing ones, inevitably causing damage to the world economy [5], [6].

In the past few years, various deep learning-based methods have been proposed to combat the threats imposed by malware and their variants, which can categorize into two branches []: feature-based detection methods and heterogeneous graph-based detection methods. Specifically, the feature-based detection methods, including Opcode+SVM [7], MalConv [8], Multimodal [9], CNN+BPNN [10], 2D-DWT [11] RNN+Max-Pooling [12], Monet [1], HTTP+SVM [13], PROVDETECTOR [14], MobiTive [15], and Gated-CNNs+Bi-LSTM [16] have focused on leveraging isolated signature or behavior features such as opcodes, API call sequence, and network traffic behavior extracted from large-scale known samples to detect malware and their variants; However, they investigate the isolated dynamic and static features, result in they are easily evaded by polymorphic variants. Motivated by the recent advances in graph neural network, the heterogeneous graph-based detection methods, including HinDroid [17], Scorpion [18], MatchGNet [19], AiDroid [20] and MG-DVD [21], concentrate on modeling the various signature and behavior entities as a heterogeneous graph to learn a comprehensive representation, which are capable of making full use of the interactive information among fine-grained malware entities, revealing the intrinsic patterns of the malware variants.

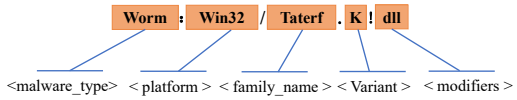

Fig. 1. Malware naming scheme.

As illustrated in Fig. 1, the universal malware naming scheme of the Computer Anti-Virus Research Organization (CARO) [22] is defined as:

```
malware_type:platform/family_name.variant!modifier,
```

which is capable of reflecting the essential characteristics and evolving of malware variants. To our best knowledge, high-quality data with human labeling could be costly, and the labeled information of the existing public malware datasets [23], [24], [21] contains at most malware_types, furthermore, the more fine-grained labels of few-shot advanced variants are expensive and time-consuming. However, the aforementioned deep learning-based detection models largely depend on a sufficient amount of input malware samples and labels pairs in a supervised manner [25], which are feeble for defending the few-shot advanced variants attacks and leads a poor generalisation for new classes.

To date, a promising solution to conquer the few-shot problem is self-supervised contrastive learning, which has shown soaring performance on representation learning in computer vision [26], [27], [28], [29], [30], natural language processing [31], [32], [33], and graph learning [34], [35], [36], [37], [38]. Commonly, the intuition of contrastive self-supervised learning is to leverage the data's inherent co-occurrence relationships as the self-supervision without the manual labeled information, which is capable of learning the matching patterns and capturing mismatching patterns via its intrinsic discriminative mechanism. Intuitively, contrastive self-supervised learning heavily depends on the elaborate instance pairs.However, most contrastive self-supervised learning has thus far focused on generating trivial data augmentations for image data or homogeneous graph-structured data and very limited work can be transferred to complex heterogeneous graph-structured data. Consequently, the existing approaches expose three major challenges of designing a self-supervised graph contrastive learning framework for advanced variant

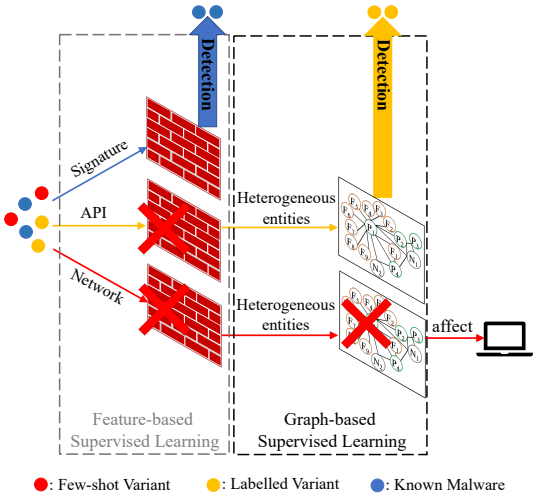

Fig. 2. Examples to illustrate the detection capabilities of different types of malware detection methods. The traditional feature-based supervised learning detection methods have ability to effectively detect the known malware denoted by the blue circles with extracting the isolated features (e.g., signature, API call sequence, and network traffic behavior), while they are incapable of identifying the in-sample variants (e.g., ubiquitous packed variants or anti-sandbox variants [2]) and the out-sample variants (e.g., advanced zero-day attacks [41] or stealthy malware variants [14]); The existing heterogeneous graph-based supervised learning detection methods have achieve remarkable success on in-sample variants denoted by the yellow circles, which attributed to their meaningful contextual interactions between fine-grained malware objects; however, the aforementioned supervised learning methods are non-transferable to few-shot out-sample variants denoted by the red circles, inevitably causing these advanced variants to violently destroy the information system by stealing sensitive data and so on.

detection on heterogeneous graph.

First, it is universally acknowledged that malware attacks have turned to be increasingly sophisticated variants, in which the majority of the variants are common packed variants [2], [39], and a percentage of the variants are advanced zero-day attacks [40], [41] or stealthy variants [14]. As shown in Fig. 2, the existing feature-based supervised learning detection methods have ability to effectively detect the known malware and the recent heterogeneous graph-based supervised learning detection methods provide a promising solution for detecting the common in-sample variants, contrarily, the few-shot advanced variants can brazenly evade the aforementioned supervised learning detection methods since they are out of the training samples and form new classes.

Second, the malware network scheme shown in Fig 3 contains both rich high-order structural information and fine-grained attribute information (semantic information). Unfortunately, the existing heterogeneous graph-based detection methods only emphasise the name of the API entities and ignore their exhaustive attribute information, resulting in they are incapable of designing comprehensive heterogeneous graph data augmentations and leading to a poor detection performance.

Third, unlike pure dependencies among the same type of nodes in homogeneous graph, the nonlinear and hierarchical heterogeneous dependencies among various system entities in heterogeneous graph. The existing data augmentations methods (e.g., ego-nets sampling [38] or uniform sampling [42]) that neglect these dependencies may still yield poor detection performance.

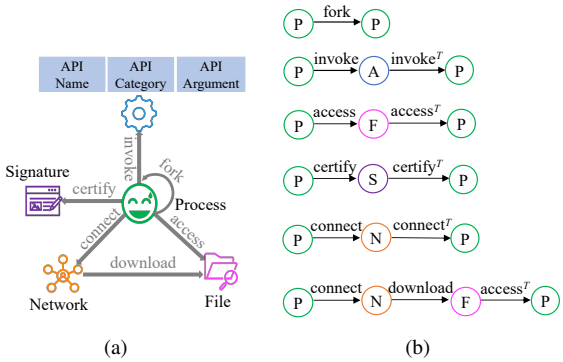

Fig. 3. (a) Network scheme of MalHGCL. (b) Meta-paths of MalHGCL.

In this work, we present a novel self-supervised Heterogeneous Graph Contrastive Learning framework for few-shot advanced Malware variant detection (MalHGCL) to address the aforementioned challenges. To summarize, our work makes the following contributions:

- We collect the latest malware and advanced variants dataset, involving hundreds of few-shot malware families, which is a few-shot malware variant dataset with the most families and has ability to more precisely distinguish polymorphic malware variants.
- We propose a novel heterogeneous graph contrastive learning framework for few-shot advanced variant detection, MalHGCL, which can effectively identify few-shot advanced variants in a self-supervised manner by learning to compare the distinguishable patterns between various malware variants instances.
- We design three types of heterogeneous graph data augmentations (i.e., API attribute masking, interaction enhancing, and meta-path sampling), each of which imposes certain semantic prior or structural prior for generating elaborate positive and negative pairs for advanced variant instance, enhancing the effectiveness of few-shot advanced variants detection and alleviating the inductive bias introduced in the training process of few-shot learning.
- We present a instance-instance discriminator, which has good generalisation by investigating the matching patterns and capturing various mismatching patterns from positive and negative instance pairs.
- We conduct ample experiments on diverse datasets to demonstrate that MalHGCL can provide competitive or superior detection performance compared with the state-of-the-art supervised detection methods, especially in few-shot advanced variant datasets.

The rest of this paper is organized as follows. Section II first introduces the preliminary definitions used throughout the paper. Section III then illustrates the methodology of our MalHGCL in detail. Section IV performs extensive experimental analyses to evaluate the few-shot advanced variant detection performance of MalHGCL. Section V reviews the related works. Finally, the conclusion is summarised in Section VI.

II. PRELIMINARIES

In this section, we introduce the preliminaries and provide major problem definitions. Accordingly, the definition of heterogeneous graph is given as follows:

***Definition** 1:* **Heterogeneous Graph of Malware Variant (HGMV).** A malware variant instance with different types of malware entities and relationships can be represented as a heterogeneous graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$ with a node type mapping $\Psi : \mathcal{V} \mapsto \mathcal{T}$ and an edge type mapping $\psi : \mathcal{E} \mapsto \mathcal{R}$. Let $\mathcal{V}$ be the set of nodes, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ be the set of relationships between nodes in $\mathcal{V}$, and $\mathbf{X} \in \mathbb{R}^{d \times f}$ is the attribute matrix. Each node $v \in \mathcal{V}$ belongs to one particular malware entity type in the node type set $\mathcal{T} : \Psi(v) \in \mathcal{T}$, and each edge $e \in \mathcal{E}$ belongs to a particular relationship type in the edge type set $\mathcal{R} : \psi(e) \in \mathcal{R}$, where $|\mathcal{T}| + |\mathcal{R}| > 2$. The i^{th} row vector $\mathbf{x}_i \in \mathbb{R}^f$ of the attribute matrix denote the attribute information of the i^{th} API node.

In this paper, HGMV investigates 5 types of malware objects involving process, API, file, network, and signature, and 6 types of interactive relationships, as shown in Figure 3. with the constructed HGMV, we formalize the few-shot advanced variants detection as a self-supervised contrastive learning classification task on the heterogeneous graph as bellow.

***Definition** 2:* **Few-shot Advanced Variants Detection Based on Heterogeneous Graph Contrastive Learning.** Given a malware variant instance heterogeneous graph $\mathcal{G}_i$, MalHGCL first randomly augments m positive instances $\boldsymbol{G}_P^i = (\mathcal{G}_{p.1}^i, \mathcal{G}_{p.2}^i, \cdots, \mathcal{G}_{p.m}^i)$ with two levels data augmentations on the original $\mathcal{G}_i$, and randomly chooses n negative instances $\boldsymbol{G}_Q^i = (\mathcal{G}_{q.1}^i, \mathcal{G}_{q.2}^i, \cdots, \mathcal{G}_{q.n}^i)$ from the rest malicious families. After that, MalHGCL learns the graph-level embeddings of $\mathcal{G}_i$, $\boldsymbol{G}_P^i$, and $\boldsymbol{G}_Q^i$ by encoders GIN_o, GIN_p, and GIN_q, respectively. Finally, the instance discriminators based on contrastive learning evaluate the agreement of each instance pair and output the predicted score of the target variant $\mathcal{G}_i$.

To capture the unique semantic information, the meta-path [43] is introduced as an interdependence sequence that connects two process objects.

***Definition** 3:* **Meta-path**. A meta-path [43] is a relationship template depicting the link types between two nodes, and is denoted in the form of $v_1 \xrightarrow{R_1} v_2 \xrightarrow{R_2} \cdots \xrightarrow{R_i} v_n$, which defines a commuting relation $R = R_1 \diamond R_2 \diamond \cdots \diamond R_n$, where $\diamond$ denotes the composition operation between node v_1 and v_n.

In this paper, 6 types of meta-paths demonstrated in Figure 3(b). Different meta-paths represent different semantic information, for example, M_2:$Process \xrightarrow{invoke} API \xrightarrow{invoke^T} Process$ depicts the semantic that "two processes invoke the same API".

***Definition** 4:* **Metapath-based explicit neighborhood and interaction-enhanced neighborhood.** Given a target process node P_{Tar} and a Pre-defined meta-path M_m in the HGMV, the metapath-based explicit neighborhood $N_{P_{Tar}}^{(m)}$ is defined as the set of all path reachable objects when P_{Tar} walks along the given meta-path M_m. Additionally, the metapath-based interaction-enhanced neighborhood $\widehat{N}_{P_{Tar}}^{(m)}$ is defined as the set of the 0-order neighbors of the explicit neighborhood $N_{P_{Tar}}^{m}$ along with their 1-order process type neighbors in HGMV.

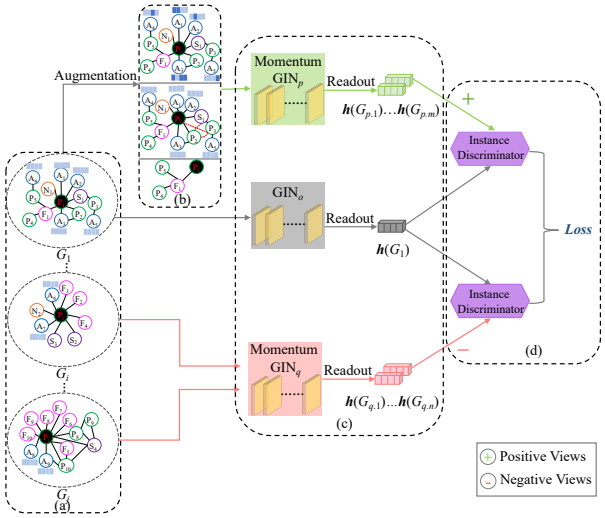

Fig. 4. The overall framework of MalHGCL. The framework is composed of four components: (a) Heterogeneous graph construction aims to model the malware objects of each variant instance with a heterogeneous graph, which holds processes, APIs, files, networks, signatures, and their interactive relationships. (b) Prior-based heterogeneous graph data augmentations generates elaborate positive and negative instance pairs by three types of transformations, including API attribute masking, interaction enhancing, and meta-path sampling. (c) GIN encoders learn the graph-level representations of $\mathcal{G}_i$, $\mathcal{G}_p$, and $\mathcal{G}_q$, respectively. (d) Discriminator-based variant detection evaluate the agreement between the non-linearly transformed representations of instance pairs via a contrastive loss and predict the target variant.

As shown in Figure 5, the explicit neighborhood based on meta-path M_m=$Process \xrightarrow{certify} Signature \xrightarrow{certify^T} Process$ is represented as $N_{P_1}^{(m)}$=$\{P_1, S_1, P_3\}$, correspondingly, the interaction-enhanced neighborhood based on meta-path M_m is represented as $\widehat{N}_{P_1}^{(m)}$=$\{P_1, S_1, P_3, P_2\}$.

III. METHODOLOGY

Inspired by the achievements of contrastive learning in few-shot classification [44], [45], we study the potential of heterogeneous graph contrastive learning model for few-shot advanced variants detection, abbreviated as MalHGCL. Concretely, given a set of heterogeneous graphs of the advanced variant instances from diverse malware families, such as "Trojan.Kazy" and "Virus.W32.MyDoom.A", we seek to leverage graph isomorphism network (GIN) based contrastive learning model to map them to different clusters. Thus, the critical questions for our MalHGCL is: (1) How to define the instances in advanced variant detection tasks? (2) How to design the positive and negative instance pairs for malware heterogeneous graph? (3) How to learn the representations of the malware instances? (4) What are the discrimination rules?

Next we exhaustively present the four components of MalHGCL framework by correspondingly answering the aforementioned questions.

A. Malware Variant Heterogeneous Graph Construction with Dense Attributes

Unlike the CV and NLP tasks to define an instance as an image or a sentence, our purpose is to purely discover the

TABLE I
API ATTRIBUTE OVERVIEW.

Attribute	Data	Detail	Dim
API name	String	Unique identifier	8
API category	String	Function and purpose	4
API argument	String	Registry keys, DLLs, and address	32

malicious attack patterns from a large number of executive behavior events, which leaves the natural choice of a heterogeneous graph as an instance, involving various malware objects and their relationships from execution events.

However, it is naive to only consider the API names and overlook their category and argument attribute information in previous heterogeneous graph-based detection works [19], [20], which loses some important information and misjudges the benign and malware. For instance, the names of two write API operations would be exactly the same, while the write API operation might be benign when the target file is created by the program itself but be malicious if the target file is a system file. Consequently, MalHGCL aims to build a heterogeneous graph with various malware objects as well as API attribute information. As shown in Table I, our API attributes consists of three types of information, involving 340 API names in total which belong to 15 categories. Moreover, we intend to extract the most important values about registry keys, file paths, DLLs, and IP addresses from API arguments.

Concretely, MalHGCL automatically extracts five types of malware objects (i.e., process, API, file, signature, and network), six types of high-level interactions (i.e., $Process \xrightarrow{fork} Process$, $Process \xrightarrow{invoke} API$, $Process \xrightarrow{access} File$, $Process \xrightarrow{certify} Signature$, $Process \xrightarrow{connect} Network$, and $Network \xrightarrow{download} File$), and three types of API attribute information (i.e., API name, API category, and API argument) from the execution events of advanced variants in Cuckoo sandbox [46], which can efficiently characterize the malware features and evolutionary patterns of advanced variants. Then we start from the target process node P_{Tar}, successively insert the interaction e into HGMV where $e.Nei \in V$. Intriguingly, to encode a sequence of API attributes strings into the fixed-length vectors, we utilizes a feature hashing trick [47] to achieve the dense attribute matrix $\boldsymbol{X}_{P_{Tar}}$, which is capable of preserving the original features and improving the detection efficiency due to the low-dimensional matrix. Formally, given an attribute string s, where each element of s is either a word or a character, the value of the t-th bin of the fixed-length attribute vector x is calculated by:

$$x_t(s) = \sum\nolimits_{g:h(s_g)=t} \xi(s_g), \tag{1}$$

where h is the hash function that maps each element in s, e.g., s_g, to an integer $u \in \{1, \cdots, U\}$ as the bin index, among them, U_{name}=8, $U_{category}$=4, and $U_{argument}$=32; ξ is another hash function that maps each element in s to $\{\pm 1\}$. Moreover, hash functions h and ξ are independent.

Eventually, we achieve the fine-grained HGMV is demonstrated in Fig. 4(a), represented as adjacency matrix $\boldsymbol{A}_{P_{Tar}}$ and attribute matrix $\boldsymbol{X}_{P_{Tar}}$.

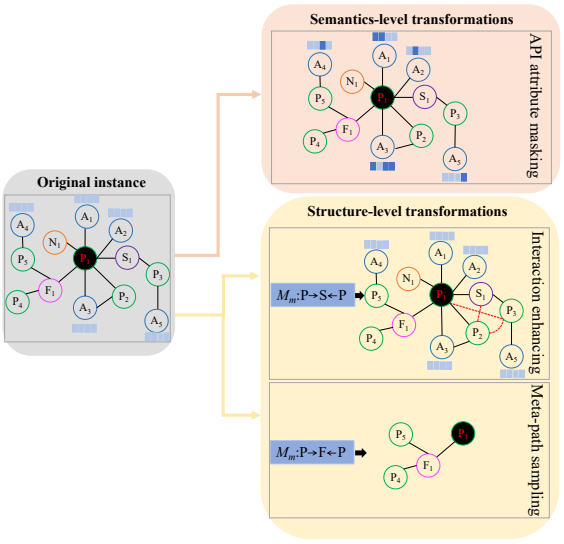

Fig. 5. Overview of data augmentations for heterogeneous graph. **Top**: API attribute masking. **Middle**: Interaction enhancing. **Bottom**: Meta-path sampling

B. Prior-based Heterogeneous Graph Data Augmentations

Recent works empirically show that congruent and incongruent instance pairs straightforwardly determine the performance of contrastive learning [48], [36]. Unlike geometric information in images or pure dependencies in homogeneous graph, creating realistically rational augmented instances of heterogeneous graph is not a trivial task, which should consider both structural information as well as attribute information.

To this end, we focus on two levels heterogeneous graph data augmentations by imposing certain semantic prior or structural prior. As shown in Figure 5, API attribute masking belongs to semantics-level augmentation, which only performs the transformations on the attribute matrix $\boldsymbol{X}$; contrarily, interaction enhancing and meta-path sampling belong to structure-level augmentations, which perform the transformations on the adjacency matrix $\boldsymbol{A}$.

1) API Attribute Masking: Given a heterogeneous graph $\mathcal{G}_i$ of i-th variant instance in the few-shot dataset, expressed as $(\boldsymbol{A}_i, \boldsymbol{X}_i)$, we formulate the transformations function $\mathcal{T}^{mask}$ as:

$$T^{mask}(\mathbf{A}_i, \mathbf{X}_i) = (\mathbf{A}_i, T^{mask}(\mathbf{X}_i)). \tag{2}$$

As shown in the top panel of Figure 5, we extend the node attribute masking [38] in the attributed networks to perform the transformation on the attribute matrix $\boldsymbol{X}_i$, which randomly masks a portion of attributes of all API nodes. Formally, we have

$$T^{mask}(\mathbf{X}_i) = \mathbf{X}_i * (1 - \mathbf{L}_m) + \mathbf{V} * \mathbf{L}_m, \tag{3}$$

where $*$ represents the element-wise multiplication; $\boldsymbol{L}_m$ denotes the masking location matrix, and $\boldsymbol{V} \sim N(\mu, \sigma^2)$ denotes the masking Gaussian noise.

Algorithm 1 Interaction Enhancing

Input: A heterogeneous graph $\mathcal{G}_i$ = ($\boldsymbol{A}_i$, $\boldsymbol{X}_i$), a meta-path M_m;

Output: Diffusive instance $\mathcal{G}'_i$ = ($\boldsymbol{A}$'$_i$, $\boldsymbol{X}_i$);

1: $T_{P_{Tar}} \leftarrow P_{Tar}$;
2: $N^{(m)}_{P_{Tar}} \leftarrow \{u|(P_{Tar}, u) \in M_m, (u, P_{Tar}) \in M_m\}$;
3: $\widehat{T_{P_{Tar}}}$=$T_{PTar} \bigcup \{u \mid u.\text{type} = \text{process} \cap (P_{Tar}, u) \in \mathcal{E}\}$;
4: **for** $n \in 1$ to $|N^{(m)}_{P_{Tar}}|$ **do**
5: $\widehat{N^{(m)}_{P_{Tar}}} = \widehat{T_{nei_1}} \bigcup \widehat{T_{nei_2}} \cdots \bigcup \widehat{T_{nei_{|N^{(m)}_{P_{Tar}}|-1}}}$;
6: **end for**
7: **for** $x \in \widehat{T_{P_{Tar}}}$ **do**
8: $r(x) = i \leftarrow x$ is the i-th node in the adjacency matrix $\boldsymbol{A}_i$;
9: **for** $y \in \widehat{N^{(m)}_{P_{Tar}}}$ **do**
10: $c(y) = j \leftarrow y$ is the j-th node in the adjacency matrix $\boldsymbol{A}_i$;
11: $\boldsymbol{A}[i][j] = 1$;
12: **end for**
13: **end for**
14: **return** $\mathcal{G}'_i$ = ($\boldsymbol{A}$'$_i$, $\boldsymbol{X}_i$)

2) Interaction Enhancing: To our best knowledge, the majority of the existing homogeneous graph data augmentations concentrate on sampling subgraphs from the original graph, which would loss several treasurable information, there is less works to explore graph diffusion to yield richer contrastive instances based on the original graph since predicting new interactions in heterogeneous graphs is challenging. Accordingly, as shown in the middle panel of Figure 5, MalHGCL investigates interaction enhancing to perform the transformation on the adjacency matrix $\boldsymbol{A}_i$, which leverages the explicit and implicit interactive information between node pairs to generate the powerful contrastive instances for self-supervised advanced variant detection.

Essentially, given a heterogeneous graph $\mathcal{G}_i$ = ($\boldsymbol{A}_i$, $\boldsymbol{X}_i$), interaction enhancing algorithm is shown in Algorithm 1, whose implementation details as follows:

step1: Searching the metapath-based explicit sets $T_{P_{Tar}}$ and $N^{(m)}_{P_{Tar}}$ of *target*-side and *neighbor*-side, respectively, which leverages the pre-defined meta-path M_m to guide the random walk for the target process node P_{Tar} (lines 1-2).

step2: Expanding the metapath-based interaction-enhanced sets $\widehat{T_{P_{Tar}}}$ and $\widehat{N^{(m)}_{P_{Tar}}}$ of *target*-side and *neighbor*-side, respectively, which concentrates on fully expanding the 1-order process neighbor nodes of each explicit node in the explicit sets $T_{P_{Tar}}$ and $N^{(m)}_{P_{Tar}}$, satisfying the trade-off between rich semantics and cost-effective speed (lines 3-6).

step3: Interacting with each other in sets $\widehat{T_{P_{Tar}}}$ and $\widehat{N^{(m)}_{P_{Tar}}}$ by transforming the value of the corresponding position in the adjacency matrix $\boldsymbol{A}_i$ to 1 (lines 7-13).

step4: Obtaining the diffusive instance $\mathcal{G}'_i$ = ($\boldsymbol{A}$'$_i$, $\boldsymbol{X}_i$) (line 14).

3) Meta-path Sampling: Considering the remarkable success of the sampling-based data augmentation on homogeneous graphs, we propose a tailored meta-path sampling for heterogeneous graphs, which instinctively captures the more discriminative malicious patterns of various heterogeneous entities. In particular, given a heterogeneous graph $\mathcal{G}_i$ = ($\boldsymbol{A}_i$, $\boldsymbol{X}_i$), the transformations function $\mathcal{T}^{sample}$ as:

$$T^{sample}(\mathbf{A}_i, \mathbf{X}_i) = (T^{sample}(\mathbf{A}_i), \mathbf{X}_i). \tag{4}$$

Concretely, meta-path sampling can sample sub-graphs based on the pre-defined meta-path M_m, as shown in the bottom panel of Figure 5, which walks starting from the target process node P_{Trag} and collects all path reachable nodes based on M_m.

Figure 5 shows three examples of augmented heterogeneous instances based on the original instance, MalHGCL treats two random augmented instance of the same heterogeneous graph as the positive instance pair, which form ($\mathcal{G}_i$, $\mathcal{G}^i_{p_m}$). Additionally, MalHGCL chooses the variant instances from the rest malicious families as the negative instances, which form ($\mathcal{G}_i$, $\mathcal{G}^i_{q_n}$).

C. GIN Encoder

The target of the graph encoders is to learn the graph-level low-dimensional representations of diverse few-shot variant instances. Technically, we utilize graph isomorphism network (GIN) [49], a powerful graph neural network, as the graph encoders in our MalHGCL, which has been empirically confirmed that it performs better than all GNNs (*e.g.*, GCN [50] and GraphSAGE [51]).

Intuitively, as described in Figure 4(c), given the original variant instance $\mathcal{G}_i$, m augmented positive instances set $\mathcal{G}^i_P$, and n sampled negative instances set $\mathcal{G}^i_Q$, we first define three identical encoders GIN$_o$, GIN$_p$, and GIN$_q$ to simultaneously produce the low-dimensional representations of the contrastive instance pairs.

1) Aggregate Node-level representation $\mathbf{h}_{P_{Tar}}$: Concretely, we first obtain the node-level representations $\mathbf{h}_{P_{Tar}}$ of the target process node P_{tar}, which is iteratively updated by combining its own feature with the aggregated representations over its neighbors. The k-th layer of node-level aggregator is:

$$\mathbf{h}^{(k)}_{P_{Tar}} = MLP^{(k)}\left((1+\epsilon^{(k)}) \cdot \mathbf{h}^{(k-1)}_{P_{Tar}} + \sum_{u \in N_{P_{Tar}}} \mathbf{h}^{(k-1)}_u\right), \tag{5}$$

where $k \in \{1, 2, \ldots, K\}$ denotes the index of the layer, $\boldsymbol{h}^{(k-1)}_{P_{Tar}}$ and $\boldsymbol{h}^{(k-1)}_u$ are the node representations of target process node P_{Tar} and neighbor node u at the *(k-1)*-th layer, respectively, and they can initialize by their state vectors. ϵ_k is a trainable parameter to balance the node-level representations of $\mathbf{h}^{(k-1)}_{P_{Tar}}$ and $\mathbf{h}^{(k-1)}_u$. Hence, the full node-level representation of P_{Tar} is:

$$\mathbf{h}_{P_{Tar}} = CONCAT([\mathbf{h}^{(k)}_{P_{Tar}}]^K_{k=1}). \tag{6}$$

2) Filter crucial node set $\boldsymbol{C}_i$: Recall that we investigate the graph-level representation $\boldsymbol{h}_{\mathcal{G}_i}$ of the variant instance heterogeneous graph $\mathcal{G}_i$, centering on the target process node

P_{Tar}, we actually consider the crucial nodes in the complete heterogeneous graph $\mathcal{G}_i$ instead of aggregating all the nodes to obtain the final graph-level representation, which is capable of enhancing the time efficiency. Formally, given the semantic-unique meta-paths $\boldsymbol{M}$, the crucial nodes set $\boldsymbol{C}_i$ in the variant heterogeneous graph instance $\mathcal{G}_i$ can be formulated as:

$$\mathbf{C}_i = \bigcup_{m\in[1,|M|]} N^{(m)}_{P_{Tar}}, \tag{7}$$

where $N^{(m)}_{P_{Tar}}$ denotes the neighbors set guided by meta-path M_m from the target process node P_{Tar}.

3) Generate graph-level representation $\boldsymbol{h}_{\mathcal{G}_i}$: Finally, we implement a summation READOUT function as the key operation to compute the graph-level representations over $\boldsymbol{C}_i$.

$$\mathbf{h}_{\mathcal{G}_i} = READOUT(\mathbf{h}_v | v \in \mathbf{C_i}). \tag{8}$$

D. Discriminator-based Variant Detection

A properly designed contrastive discriminator related to variant detection should concentrated on multiple different positive and negative instances, which would achieve the few-shot malware variants detection by progressively learn the "distinguishable" information to discriminate different instances. Specifically, our MalHGCL contrasts the comprehensive representations of the two heterogeneous graphs in an instance pair, as shown in Figure 4(d), it aims to capture the similarity between the target variant instance $\mathcal{G}_i$ and the augmented positive instance $\mathcal{G}^i_{p.m}$, simultaneously, capture the dissimilarity between the target variant instance $\mathcal{G}_i$ and the sampled negative instance $\mathcal{G}^i_{q.n}$.

Inspired by the success of the projection heads in SimCLR [29], the discriminator-based variant detection first employs a non-linear projection head to the graph-level representation $\boldsymbol{h}_{\mathcal{G}_i}$ before computing the pair-wise similarity. Formally, we view the non-linear projection head as:

$$\mathbf{z}_{\mathcal{G}_i} = MLP(\mathbf{h}_{\mathcal{G}_i}). \tag{9}$$

Given a set of variant instances, MalHGCL aims to identify the unique pair of positive instances or the pair of negative instances. Concretely, on the one hand, we maintain a mini-batch B^1_i of n negative instances ($\mathcal{G}^i_{q.1}$, $\mathcal{G}^i_{q.2}$, $\cdots$, $\mathcal{G}^i_{q.n}$) by randomly sampling from the rest malicious families and a positive instance $\mathcal{G}^i_p$. From a dictionary look-up perspective, we intend to look up the single instance (denoted by $\mathcal{G}^i_p$) that the target variant instance $\mathcal{G}_i$ matches in the (n+1)-size B^1_i. Then the loss function InfoNCE [27] for the unique positive pair of ($\mathcal{G}_i$, $\mathcal{G}^i_p$) is defined as:

$$l_{i,p} = \frac{exp(sim(\mathbf{z}_{\mathcal{G}_i}, \mathbf{z}_{\mathcal{G}^i_p})/\tau)}{exp(sim(\mathbf{z}_{\mathcal{G}_i}, \mathbf{z}_{\mathcal{G}^i_p})/\tau) + \sum_{d=1}^{n} exp(sim(\mathbf{z}_{\mathcal{G}_i}, \mathbf{z}_{\mathcal{G}^i_{q.d}})/\tau)}, \tag{10}$$

where $\boldsymbol{z}_{\mathcal{G}_i}$, $\boldsymbol{z}_{\mathcal{G}^i_p}$, and $\boldsymbol{z}_{\mathcal{G}^i_{q.d}}$ denote the representations of $\mathcal{G}_i$, $\mathcal{G}^i_p$, and $\mathcal{G}^i_{q.d}$ after nonlinear transformation. τ denotes a temperature parameter.

On the other hand, we as well as maintain a mini-batch B^2_i of m positive instances ($\mathcal{G}^i_{p.1}$, $\mathcal{G}^i_{p.2}$, $\cdots$, $\mathcal{G}^i_{p.m}$) by data augmentations of the target variant instance $\mathcal{G}_i$ and a negative instance $\mathcal{G}^i_q$. Correspondingly, we also look up the single instance (denoted by $\mathcal{G}^i_q$) from the (m+1)-size B^2_i, and the loss function for the unique negative pair of ($\mathcal{G}_i$, $\mathcal{G}^i_q$) is defined as:

$$l_{i,q} = \frac{exp(sim(\mathbf{z}_{\mathcal{G}_i}, \mathbf{z}_{\mathcal{G}^i_q})/\tau)}{exp(sim(\mathbf{z}_{\mathcal{G}_i}, \mathbf{z}_{\mathcal{G}^i_q})/\tau) + \sum_{d=1}^{m} exp(sim(\mathbf{z}_{\mathcal{G}_i}, \mathbf{z}_{\mathcal{G}^i_{p.d}})/\tau)}. \tag{11}$$

The final loss L is computed across all positive pairs and all negative pairs, which is formally expressed as:

$$\mathcal{L} = \frac{1}{m}\sum_{p=1}^{m} l_{i,p} - \frac{1}{n}\sum_{q=1}^{n} l_{i,q}. \tag{12}$$

Essentially, we maximize the contrastive loss to train the encoders GIN_o, GIN_p, GIN_q, and the projection head $\boldsymbol{z}(\cdot)$. Recall that we adopt multiple trainable encoders to generate the representations of the instances in the mini-batch B_i, which would bring out extremely computational overhead.

Algorithm 2 The Overall Procedure of MalHGCL

Input: A heterogeneous graph of variant instance$_i$:$\mathcal{G}_i$ = ($\boldsymbol{A}_i$, $\boldsymbol{X}_i$), Augmentations: $\mathcal{T}^{mask}$, $\mathcal{T}^{enhance}$, $\mathcal{T}^{sample}$, Encoders: GIN_o, GIN_p, GIN_q, Projection heads: MLP_o, MLP_p, MLP_q, m, n;
Output: Trained GIN_o, GIN_p, GIN_q, MLP_o, MLP_p, MLP_q;

$\boldsymbol{G}^i_P = (\mathcal{G}^i_{p.1}, \mathcal{G}^i_{p.2}, \cdots, \mathcal{G}^i_{p.m}) = \mathcal{T}^{mask}(\mathcal{G}_i) \cup \mathcal{T}^{enhance}(\mathcal{G}_i) \cup \mathcal{T}^{sample}(\mathcal{G}_i)$;
$\boldsymbol{G}^i_Q = (\mathcal{G}^i_{q.1}, \mathcal{G}^i_{q.2}, \cdots, \mathcal{G}^i_{q.n})$=sampling the negative instances from the rest malicious families;
for $\{\mathcal{G}^i_{p.a}\}^m_{a=1} \in \boldsymbol{G}^i_P$ **do**
  $B^1_i = \mathcal{G}^i_{p.a} \cup \boldsymbol{G}^i_Q$;
  $\mathbf{h}_{\mathcal{G}_i} = GIN_o(\mathcal{G}_i)$;
  $\mathbf{z}_{\mathcal{G}_i} = MLP_o(\mathbf{h}_{\mathcal{G}_i})$;
  $\mathbf{h}_{\mathcal{G}^i_{p.a}} = GIN_p(\mathcal{G}^i_{p.a})$;
  $\mathbf{z}_{\mathcal{G}^i_{p.a}} = MLP_p(\mathbf{h}_{\mathcal{G}^i_{p.a}})$;
  for $\{\mathcal{G}^i_{q.d}\}^n_{d=1} \in B^1_i \backslash \mathcal{G}^i_{p.a}$ **do**
    $\mathbf{h}_{\mathcal{G}^i_{q.d}} = GIN_q(\mathcal{G}^i_{q.d})$;
    $\mathbf{z}_{\mathcal{G}^i_{q.d}} = MLP_q(\mathbf{h}_{\mathcal{G}^i_{q.d}})$;
  end for
  Computing $l_{i,p}$ by using *Eq.* 10;
end for
for $\{\mathcal{G}^i_{q.b}\}^n_{b=1} \in \boldsymbol{G}^i_Q$ **do**
  $B^2_i = \mathcal{G}^i_{q.b} \cup \boldsymbol{G}^i_P$;
  $\mathbf{h}_{\mathcal{G}^i_{q.b}} = GIN_q(\mathcal{G}^i_{q.b})$;
  $\mathbf{z}_{\mathcal{G}^i_{q.b}} = MLP_q(\mathbf{h}_{\mathcal{G}^i_{q.b}})$;
  for $\{\mathcal{G}^i_{p.d}\}^m_{d=1} \in B^2_i \backslash \mathcal{G}^i_{q.b}$ **do**
    $\mathbf{h}_{\mathcal{G}^i_{p.d}} = GIN_p(\mathcal{G}^i_{p.d})$;
    $\mathbf{z}_{\mathcal{G}^i_{p.d}} = MLP_p(\mathbf{h}_{\mathcal{G}^i_{p.d}})$;
  end for
  Computing $l_{i,q}$ by using *Eq.* 11;
end for
for a=1 to m and b=1 to n **do**
  $\mathcal{L} = \frac{1}{m}\sum_{p=1}^{m} l_{i,p} - \frac{1}{n}\sum_{q=1}^{n} l_{i,q}$;
end for
Updating θ_i by maximizing $\mathcal{L}$ with backpropagation;
$\theta_p = \lambda * \theta_p + (1-\lambda) * \theta_i$;
$\theta_q = \lambda * \theta_q + (1-\lambda) * \theta_i$;
return GIN_o, GIN_p, GIN_q, MLP_o, MLP_p, MLP_q

Consequently, we wisely leverage an economical momentum strategy [28] to achieves a larger dictionary size. Concretely, when optimizes the parameters of multiple encoders, such as GIN_o, $GIN_{p.1}$, $\cdots$, $GIN_{p.m}$, we only train the parameters θ_i of GIN_o by backpropagation, and then momentum-based update the rest parameters of $GIN_{p.1}$, $\cdots$, $GIN_{p.m}$ by utilizing the θ_i.

IV. Experiments

A. Datasets

B. Experimental Settings

C. Few-shot Variant Detection Results

D. Parameter Sensitivity

E. Ablation Study

F. Visualization

We collect the latest malware and advanced variants dataset, involving hundreds of few-shot malware families, which is a few-shot malware variant dataset with the most families and has ability to more precisely distinguish polymorphic malware variants.

V. Related Work

In this section, we review the most related work of malware variant detection and graph contrastive learning.

A. Malware Variant Detection

Feature-based Supervised Learning Malware variant Detection. Feature-based supervised learning malware variant detection has been extensively studied in past years due to the success of machine learning and deep learning in handling a large number of labeled samples, its goal is to extract representative features to detect malware variants. More specifically, these feature-based supervised learning detection methods can be roughly divided into static signature-based [7], [8], [9], [10], [11] and dynamic behavior-based [12], [1], [13], [15], [16].

Particularly, Gaviria *et al.* [7] implemented Support Vector Machine (SVM) to tackle the frequency information of the opcode. Nevertheless, these opcode feature is too single and noisy to contain enough information to provide acceptable detection accuracy. Raff *et al.* [8] utilized Convolutional Neural Networks (CNN) to investigate the raw byte sequence of malware, which is the first architecture that can successfully process long sequence of over two million steps. However, the accuracy of MalConv is limited due to the raw byte sequence contained a lot of noise; Kim *et al.* [9] presented a multimodal malware detection framework based on various static features, where different types of the static feature are inputted in different neural networks separately, and then the generated feature vectors of each initial network is connected to the final deep neural network (DNN) to produce the binary classification results. Considering the trade-off between detection accuracy and time efficiency, Zhang *et al.* [10] chose the most critical two types of static feature – opcode and API from various static features. They adopted Convolutional Neural Networks (CNN) and Back Propagation Neural Network (BPNN) to extract high-level features from opcode and API calls, respectively, which significantly improved the detection accuracy. Chawla *et al.* [11] proposed a signal processing methodology to remotely detect malware by leveraging the temporal and spectral features from Electromagnetic (EM)-traces. The system first used Discrete Wavelet Transform (DWT) to extract feature from EM-side, and then performed Support Vector Machine (SVM) or Random Forest (RF) to detect malware. Unfortunately, the majority of signature features of malware are easily confused by the obfuscation and transformation technologies. To mitigate the problems, Pascanu *et al.* [12] aimed at extracting the executed API sequences from the isolated Cuckoo [46] and training a Recurrent Neural Network (RNN) to identify malware. However, the method obtained an inferior performance when the length of the API sequence is larger than the specified threshold. Sun *et al.* [1] combined "static signature" with "runtime behavior" to detect malware variants, which leverages the static structure and the runtime behaviors of a new malware variant and its earlier generation are usually similar, and has ability to defend against 10 types of obfuscation and transformation techniques. Wang *et al.* [13] observed that the network traffic generated by malware can reveal the attack intention of malware, hence, they considered each HTTP flow as a text document, which can be handled by natural language processing (NLP). They built a Support Vector Machine (SVM) classifier for malware detection and achieved an excellent detection performance. To mitigate the time-consuming issue, Feng *et al.* [15] proposed an effective malware detection system MobiTive, which designed a feature updating method to extract behavior features, and then fed into a customized deep neural network to provide a real-time detection. Zhang *et al.* [16] pointed out the detection effectiveness of the existing dynamic detection methods is limited by the trivial API name; accordingly, they presented a novel feature extraction approach to mine more abundant information from API arguments, which can be processed by a hybrid model.

However, the majority of aforementioned feature-based supervised learning detection methods overuse the isolated signature or behavior features to detect the malware variants, they ignore the interactive relationships among various malware objects and cannot learn the evolutionary patterns between malware and their variants; what's more, they are vulnerable to different obfuscation and anti-sandbox techniques, resulting in a unpromising landscape identifying advanced variants (zero-day attacks or stealthy malware variants). To address these problems, our proposed MalHGCL introduces the heterogeneous graphs to model the interactive relationships among different types malware objects from the execution streams, which is capable of effectively capturing the inherent interactions between malware objects and combating the popular transformation techniques.

Heterogeneous Graph-based Supervised Learning Malware variant Detection. With the rocketing growth of the heterogeneous graph, several heterogeneous graph-based supervised learning approaches are presented to malware variant detection. Hou *et al.* [17] first adopted structured hetero-

geneous information network (HIN) for malware detection. They extracted two types of heterogeneous entity (i.e., Android application (app) and API) and their relationships to construct a heterogeneous information network, then they designed a similarity-based approach to detect the malware. Fan *et al.* [18] proposed Metagraph2vec, which simultaneously extracted content-based features and relation-based features from constructed heterogeneous graph. The system implemented skip-gram [50] to generate the graph embedding, which has been proven to be inferior than 2-layer Graph Convolutional Networks (GCN) on many graph data. Wang *et al.* [19] designed MatchGNet, where the similarities between the target malware and all benign samples can be measured by leveraging metapath-based graph representations. Ye *et al.* [20] presented an in-sample node embedding – HGiNE and an out-sample node embedding – HG2Img, respectively, which adopted the in-sample node embedding in the large-scale heterogeneous graph to learn the representations of out-sample nodes. Nevertheless, dynamic graph learning for real-time malware detection remains under-explored; thus, Liu *et al.* [21] attempt to investigate the real-time detection framework based on the sliding window, which can effectively and efficiently defense the malware attacks with two dynamic walk-based heterogeneous graph learning methods.

However, the aforementioned heterogeneous graph-based supervised learning detection methods heavily depend on large corpus of labeled data, and fail to identify few-shot advanced variants due to poor generalisation for new target classes, inevitably causing these advanced variants to violently destroy the information system. Contrarily, this paper presents MalHGCL, introducing contrastive Learning to recognize the few-shot advanced variants in a self-supervised manner, which has good generalisation performance by investigating the inherent and discriminative patterns of various malware family instances.

Self-supervised Learning Malware variant Detection. To cope with the problem of one-shot malware outbreak, Park *et al.* [52] developed the generative adversarial auto-encoder (AAE) [53] to malware detection, which detects the malware in the self-supervised manner by calculating the adversarial loss between the original malware and the adversarial samples from the Gaussian distribution.

However, there are two limitations of this method. On the one hand, they only processed the isolated API features from the custom sandbox and overlooked the plentiful interactions of various malware objects, which leads to high false positives since several variants always hide attacks in a large number of benign behaviors as well as have slightly different from benign software; On the other hand, the generative model heavily concentrate on sufficient detail of the sample, which would cause a poor detection performance once the generated adversarial sample would not contain enough information for matching. Consequently, our proposed MalHGCL designs heterogeneous graph contrastive learning to conquer the aforementioned limitations in few-shot malware detection, which constructs the malware family heterogeneous graphs to model the interactive relationships among different types of malware objects and has ability to accurately identify various polymorphic variants. Additionally, MalHGCL is capable of recognizing different advanced variants by investigating only the approximate discriminative patterns instead of sufficient detail.

B. Graph Contrastive Learning

The purpose of graph contrastive learning is to define a pre-training task to capture the dependencies between different views and achieve the classification of unlabeled graph data. It is universally acknowledged that there are three factors in it: graph data augmentations, graph encoder, and objective function [25], [42].

Local-global Contrast. The local-global contrast focuses on establishing the belonging relationship between the local feature of a sample and its global context representation. DGI [34] first attempted to extend the Deep InfoMax in the Computer Version (CV) to the graph data, which maximizes the mutual information (MI) between input and output to identify whether the learned representation is the unique feature of the sample. More specifically, DGI implemented graph convolutional network (GCN) as the encoder and generated the negative samples by destroying the original context. Unlike DGI targets learning node-level representation, Sun *et al.* [35] intended to maximize the mutual information between graph-level representations, which also leverages graph convolutional network (GCN) to learn the graph-level representation. Inspired by CMC has done to improve Deep InfoMax in Computer Version (CV), Hassani *et al.* [36] presented a contrastive multi-view representation learning method for graph data, which adopts graph diffusion to yield positive sample pairs.

However, a significant focus in local-global contrast is to explore unique and distinguishable local features based on context, which is a tremendous challenge in practical applications. Accordingly, in this paper, we design an instance-instance contrastive learning, which purposes to study the relationships between different malware family instance-level representations and has ability to directly capture the inherent patterns of different family instance.

Global-global Contrast. Global-global contrast, i.e., instance-instance contrast, rather than local-global contrast, is more crucial for a wide range of graph classification tasks. Qiu *et al.* [37] first exploited instance discrimination for graph pre-training. They utilized random walks with restart (RWR) to generate independent subgraph as instance and calculated InfoNCE [27] loss, which is friendly to large-scale graphs. Experimental results confirm that GCC has learned better transferable structural knowledge than previous works. You *et al.* [38] believed that node neighborhood reconstruction belongs to local-global contrast, and over-emphasis on neighborhood information will destroy structural information. Therefore, they proposed four augmentation methods based on homogeneous graph to produce the positive samples, which achieves better performance on dissimilar datasets.

However, the existing data augmentation methods of the aforementioned instance-instance contrastive learning only appropriate for homogeneous graphs, which ignores hetero-

geneous entities and relationships in heterogeneous graph-based advanced variants detection, resulting in a poor performance. To address this problem, this paper proposes three types of heterogeneous graph data augmentation methods (i.e., API attribute masking, interaction enhancing, and meta-path sampling), each of which imposes certain semantic prior or structural prior for generating adequate positive pairs for advanced variant instance, enhancing the effectiveness of few-shot advanced variants detection.

VI. Conclusion

References

[1] Mingshen Sun, Xiaolei Li, John CS Lui, Richard TB Ma, and Zhenkai Liang. Monet: a user-oriented behavior-based malware variants detection system for android. *IEEE Transactions on Information Forensics and Security*, 12(5):1103–1112, 2016.

[2] Xiao Chen, Chaoran Li, Derui Wang, Sheng Wen, Jun Zhang, Surya Nepal, Yang Xiang, and Kui Ren. Android hiv: A study of repackaging malware for evading machine-learning detection. *IEEE Transactions on Information Forensics and Security*, 15:987–1001, 2019.

[3] Amir Afianian, Salman Niksefat, Babak Sadeghiyan, and David Baptiste. Malware dynamic analysis evasion techniques: A survey. *ACM Computing Surveys (CSUR)*, 52(6):1–28, 2019.

[4] AvTest. Malware statistics, 2020. https://www.av-test.org/en/statistics/malware/.

[5] Luca Caviglione, Michał Choraś, Igino Corona, Artur Janicki, Wojciech Mazurczyk, Marek Pawlicki, and Katarzyna Wasielewska. Tight arms race: Overview of current malware threats and trends in their detection. *IEEE Access*, 2020.

[6] Chunlin Xiong, Tiantian Zhu, Weihao Dong, Linqi Ruan, Runqing Yang, Yan Chen, Yueqiang Cheng, Shuai Cheng, and Xutong Chen. Conan: A practical real-time apt detection system with high accuracy and efficiency. *IEEE Transactions on Dependable and Secure Computing*, 2020.

[7] José Gaviria de la Puerta and Borja Sanz. Using dalvik opcodes for malware detection on android. *Logic Journal of the IGPL*, 25(6):938–948, 2017.

[8] Edward Raff, Jon Barker, Jared Sylvester, Robert Brandon, Bryan Catanzaro, and Charles K Nicholas. Malware detection by eating a whole exe. In *Workshops at the Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[9] TaeGuen Kim, BooJoong Kang, Mina Rho, Sakir Sezer, and Eul Gyu Im. A multimodal deep learning method for android malware detection using various features. *IEEE Transactions on Information Forensics and Security*, 14(3):773–788, 2018.

[10] Jixin Zhang, Zheng Qin, Hui Yin, Lu Ou, and Kehuan Zhang. A feature-hybrid malware variants detection using cnn based opcode embedding and bpnn based api embedding. *Computers & Security*, 84:376–392, 2019.

[11] Nikhil Chawla, Harshit Kumar, and Saibal Mukhopadhyay. Machine learning in wavelet domain for electromagnetic emission based malware analysis. *IEEE Transactions on Information Forensics and Security*, 16:3426–3441, 2021.

[12] Razvan Pascanu, Jack W Stokes, Hermineh Sanossian, Mady Marinescu, and Anil Thomas. Malware classification with recurrent networks. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1916–1920. IEEE, 2015.

[13] Shanshan Wang, Qiben Yan, Zhenxiang Chen, Bo Yang, Chuan Zhao, and Mauro Conti. Detecting android malware leveraging text semantics of network flows. *IEEE Transactions on Information Forensics and Security*, 13(5):1096–1109, 2017.

[14] Qi Wang, Wajih Ul Hassan, Ding Li, Kangkook Jee, Xiao Yu, Kexuan Zou, Junghwan Rhee, Zhengzhang Chen, Wei Cheng, Carl A Gunter, et al. You are what you do: Hunting stealthy malware via data provenance analysis. In *NDSS*, 2020.

[15] Ruitao Feng, Sen Chen, Xiaofei Xie, Guozhu Meng, Shang-Wei Lin, and Yang Liu. A performance-sensitive malware detection system using deep learning on mobile devices. *IEEE Transactions on Information Forensics and Security*, 16:1563–1578, 2020.

[16] Zhaoqi Zhang, Panpan Qi, and Wei Wang. Dynamic malware analysis with feature engineering and feature learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 1210–1217, 2020.

[17] Shifu Hou, Yanfang Ye, Yangqiu Song, and Melih Abdulhayoglu. Hindroid: An intelligent android malware detection system based on structured heterogeneous information network. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1507–1515, 2017.

[18] Yujie Fan, Shifu Hou, Yiming Zhang, Yanfang Ye, and Melih Abdulhayoglu. Gotcha-sly malware! scorpion a metagraph2vec based malware detection system. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 253–262, 2018.

[19] Shen Wang and S Yu Philip. Heterogeneous graph matching networks: Application to unknown malware detection. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 5401–5408. IEEE, 2019.

[20] Yanfang Ye, Shifu Hou, Lingwei Chen, Jingwei Lei, Wenqiang Wan, Jiabin Wang, Qi Xiong, and Fudong Shao. Out-of-sample node representation learning for heterogeneous graph in real-time android malware detection. In *28th International Joint Conference on Artificial Intelligence (IJCAI)*, 2019.

[21] Chen Liu, Bo Li, Jun Zhao, Ming Su, and Xu-Dong Liu. Mg-dvd: A real-time framework for malware variant detection based on dynamic heterogeneous graph learning. In Zhi-Hua Zhou, editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 1512–1519. International Joint Conferences on Artificial Intelligence Organization, 8 2021. Main Track.

[22] Vesselin Bontchev Alan Solomon, Fridrik Skulason. Computer antivirus research organization malware naming scheme, 1991. http://www.caro.org/.

[23] Angelo Schranko de Oliveira and Renato José Sassi. Behavioral malware detection using deep graph convolutional neural networks. 2019.

[24] Ferhat Ozgur Catak and Ahmet Faruk Yazı. A benchmark api call dataset for windows pe malware classification. *arXiv preprint arXiv:1905.01999*, 2019.

[25] Xiao Liu, Fanjin Zhang, Zhenyu Hou, Li Mian, Zhaoyu Wang, Jing Zhang, and Jie Tang. Self-supervised learning: Generative or contrastive. *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[26] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. *arXiv preprint arXiv:1808.06670*, 2018.

[27] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[28] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738, 2020.

[29] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[30] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent: A new approach to self-supervised learning. *arXiv preprint arXiv:2006.07733*, 2020.

[31] Lingpeng Kong, Cyprien de Masson d'Autume, Wang Ling, Lei Yu, Zihang Dai, and Dani Yogatama. A mutual information maximization perspective of language representation learning. *arXiv preprint arXiv:1910.08350*, 2019.

[32] Wenhan Xiong, Jingfei Du, William Yang Wang, and Veselin Stoyanov. Pretrained encyclopedia: Weakly supervised knowledge-pretrained language model. *arXiv preprint arXiv:1912.09637*, 2019.

[33] Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*, 2020.

[34] Petar Veličković, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax. *arXiv preprint arXiv:1809.10341*, 2018.

[35] Fan-Yun Sun, Jordan Hoffmann, Vikas Verma, and Jian Tang. Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. *arXiv preprint arXiv:1908.01000*, 2019.

[36] Kaveh Hassani and Amir Hosein Khasahmadi. Contrastive multi-view representation learning on graphs. In *International Conference on Machine Learning*, pages 4116–4126. PMLR, 2020.

[37] Jiezhong Qiu, Qibin Chen, Yuxiao Dong, Jing Zhang, Hongxia Yang, Ming Ding, Kuansan Wang, and Jie Tang. Gcc: Graph contrastive coding

for graph neural network pre-training. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1150–1160, 2020.

[38] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph contrastive learning with augmentations. *Advances in Neural Information Processing Systems*, 33:5812–5823, 2020.

[39] Kaiming Xiao, Cheng Zhu, Junjie Xie, Yun Zhou, Xianqiang Zhu, and Weiming Zhang. Dynamic defense strategy against stealth malware propagation in cyber-physical systems. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, pages 1790–1798. IEEE, 2018.

[40] Prakash Mandayam Comar, Lei Liu, Sabyasachi Saha, Pang-Ning Tan, and Antonio Nucci. Combining supervised and unsupervised learning for zero-day malware detection. In *2013 Proceedings IEEE INFOCOM*, pages 2022–2030. IEEE, 2013.

[41] Sahar Abdelnabi, Katharina Krombholz, and Mario Fritz. Visualphishnet: Zero-day phishing website detection by visual similarity. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 1681–1698, 2020.

[42] Yaochen Xie, Zhao Xu, Jingtun Zhang, Zhengyang Wang, and Shuiwang Ji. Self-supervised learning of graph neural networks: A unified review. *arXiv preprint arXiv:2102.10757*, 2021.

[43] Yizhou Sun, Jiawei Han, Xifeng Yan, Philip S Yu, and Tianyi Wu. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *Proceedings of the VLDB Endowment*, 4(11):992–1003, 2011.

[44] Zhenguo Li, Fengwei Zhou, Fei Chen, and Hang Li. Meta-sgd: Learning to learn quickly for few-shot learning. *arXiv preprint arXiv:1707.09835*, 2017.

[45] Yaqing Wang, Quanming Yao, James T Kwok, and Lionel M Ni. Generalizing from a few examples: A survey on few-shot learning. *ACM Computing Surveys (CSUR)*, 53(3):1–34, 2020.

[46] Cuckoosandbox. Cuckoo sandbox - automated malware analysis[eb/ol]. https://cuckoosandbox.org/, 2016.

[47] Kilian Weinberger, Anirban Dasgupta, John Langford, Alex Smola, and Josh Attenberg. Feature hashing for large scale multitask learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 1113–1120, 2009.

[48] Philip Bachman, R Devon Hjelm, and William Buchwalter. Learning representations by maximizing mutual information across views. *arXiv preprint arXiv:1906.00910*, 2019.

[49] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

[50] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[51] William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 1025–1035, 2017.

[52] Sean Park, Iqbal Gondal, Joarder Kamruzzaman, and Leo Zhang. One-shot malware outbreak detection using spatio-temporal isomorphic dynamic features. In *2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, pages 751–756. IEEE, 2019.

[53] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015.