Heterogeneous Temporal Graph Transformer: An Intelligent System for Evolving Android Malware Detection

Yujie Fan[1], Mingxuan Ju[1], Shifu Hou[1], Yanfang Ye[1*]
Wenqiang Wan[2], Kui Wang[2], Yinming Mei[2], Qi Xiong[2]
[1] Dept. of Computer and Data Sciences, Case Western Reserve University; [2] Tencent Security Lab, Tencent
[1] Ohio, United States; [2] Guangdong, China
{yxf370,mxj255,sxh1055,yanfang.ye}@case.edu,{johnnywan,flashwang,yinmingmei,keonxiong}@tencent.com

ABSTRACT

The explosive growth and increasing sophistication of Android malware call for new defensive techniques to protect mobile users against novel threats. To address this challenge, in this paper, we propose and develop an intelligent system named Dr.Droid to jointly model malware propagation and evolution for their detection *at the first attempt*. In Dr.Droid, we first exploit higher-level semantic and social relations within the ecosystem (e.g., app-market, app-developer, market-developer relations etc.) to characterize app propagation patterns; and then we present a structured heterogeneous graph to model the complex relations among different types of entities. To capture malware evolution, we further consider the temporal dependence and introduce a heterogeneous temporal graph to jointly model malware propagation and evolution by considering heterogeneous spatial dependencies with temporal dimensions. Afterwards, we propose a novel heterogeneous temporal graph transformer framework (denoted as HTGT) to integrate both spatial and temporal dependencies while preserving the heterogeneity to learn node representations for malware detection. Specifically, in our proposed HTGT, to preserve the heterogeneity, we devise a heterogeneous spatial transformer to derive heterogeneous attentions over each node and edge to learn dedicated representations for different types of entities and relations; to model temporal dependencies, we design a temporal transformer into the HTGT to attentively aggregate its historical sequences of a given node (e.g., app); the two transformers work in an iterative manner for representation learning. Promising experimental results based on the large-scale sample collections from anti-malware industry demonstrate the performance of Dr.Droid, by comparison with state-of-the-art baselines and popular mobile security products.

CCS CONCEPTS

• **Security and privacy → Malware and its mitigation**; • **Computing methodologies → Learning latent representations**.

*Corresponding Author.

KDD '21, August 14–18, 2021, Virtual Event, Singapore

ACM ISBN 978-1-4503-8332-5/21/08…$15.00
https://doi.org/10.1145/3447548.3467168

KEYWORDS

Heterogeneous Temporal Graph, Transformer, Malware Detection.

ACM Reference Format:
Yujie Fan[1], Mingxuan Ju[1], Shifu Hou[1], Yanfang Ye[1] and Wenqiang Wan[2], Kui Wang[2], Yinming Mei[2], Qi Xiong[2]. 2021. Heterogeneous Temporal Graph Transformer: An Intelligent System for Evolving Android Malware Detection. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '21), August 14–18, 2021, Virtual Event, Singapore.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3447548.3467168

1 INTRODUCTION

Thanks to the mobility and ever expanding capabilities, mobile devices connected to the Internet become ubiquitous, which have transformed our daily lives in various aspects (e.g., socializing, online banking, healthcare, education, daily work). Today, society's overwhelming reliance on the complex yet increasingly connected devices makes their security more important than ever. Android, as an open source and customizable operating system (OS) for mobile devices, is currently dominating the market by 71.99% [23]. However, due to its large market share and open source ecosystem of development, Android attracts not only the developers for producing legitimate Android applications (apps), but also attackers to disseminate malware (***mal***icious soft***ware***) that deliberately fulfills the harmful intent to mobile device users. Driven by considerable profits, there has been explosive growth of Android malware - i.e., according to AV-Test, the total new Android malware (including potentially unwanted apps) reached 8.81 million in 2020 [2]. The large volume of increasingly sophisticated Android malware has posed serious threats to mobile device users, such as stealing user credentials, pushing unwanted apps or advertisements (ads), blocking user accesses until ransoms are paid [12]. In particular, as a large population transitioned to work-from-home models and businesses rapidly shifted to remote operations during the global pandemic, malware attacks also pivoted. According to PurpleSec, the estimated cost of ransomware attacks nearly doubled in 2020 reaching $20 billion [21]. Therefore, the detection of evolving Android malware is of great importance to both researchers and security industry.

To protect legitimate users against Android malware attacks, the most significant line is mobile security products. However, in the never-ending arms race, attackers always devise new tactics to evade the detection. For example, malware may always adopt techniques such as repackaging and obfuscation to bypass the detection. As defenders, *our thesis* is that, no matter how a malware mutates and evolves, the goal is to evade the detection while preserving its dedicated malicious intents and functionalities. To that end, as

shown in Figure 1.(a), we propose to learn the evolution patterns for the abstraction of latent malice encoded in malware variants for their detection. In addition to evolution patterns, malware propagation also plays an important role for the detection. For example, the "TigerEyeing" trojan, a kind of Command and Control (C&C) malware [35], camouflages as legitimate apps and only executes to perform the profitable tasks on-demand; as such, merely evolution patterns characterized by app content may not be sufficient for its detection. To combat such a sophisticated malware, as illustrated in Figure 1.(b), its propagation patterns - i.e., how it's disseminated within the ecosystem (e.g., who owned/developed the app, which markets the app was distributed, etc.) - can provide complementary knowledge for its detection. To this end, as shown in Figure 1.(c), we ask ourselves *"can we jointly model malware propagation and evolution to learn their latent representations for the detection"?*

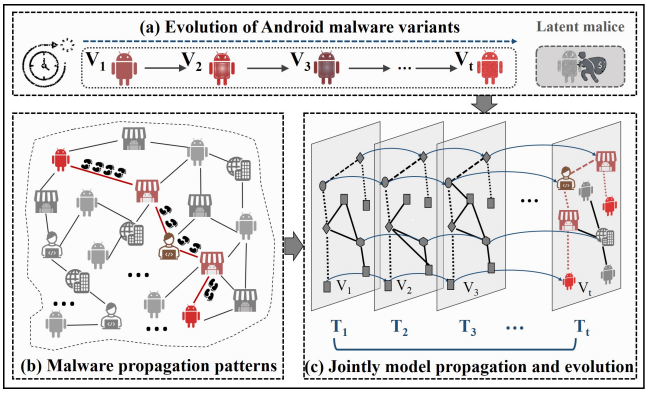


Figure 1: Motivation: jointly model propagation & evolution.

To solve the above problem, there are two major challenges: *The first one* is how to devise an integral model that is capable of depicting the complex semantic and social relations within the ecosystem (e.g., app-market, app-developer, market-developer relations etc.) to characterize app propagation patterns while considering their temporal dependencies (i.e., app evolution). *The second one* is how to design an effective framework that is able to seamlessly integrate both heterogeneous spatial dependencies and temporal dimensions for app representation learning to detect the malicious ones.

In this paper, to address the above challenges, we propose and develop an intelligent system named Dr.Droid (shown in Figure 2) for evolving Android malware detection. In Dr.Droid, to tackle the first challenge, as illustrated in Figure 2.(a.1), besides content-based features, we propose to consider higher-level semantic and social relations among apps and other types of entities (e.g., app markets, owners, names, and developers); and then we introduce a structured heterogeneous graph (HG) to model the complex relations in a comprehensive manner (shown in Figure 2.(a.2)); afterwards, to further integrate the temporal dependencies in the HGs (e.g., dynamics of evolving malware variants), we present a heterogeneous temporal graph (HTG) for modeling (illustrated in Figure 2.(a.3)). For the second challenge, to learn node representations over the HTG, the dynamics of HTG with evolution in time and space entailed the heterogeneity open a new research area from currently considered homogeneous graph structures [4, 20, 31, 38]. In this work, as shown in Figure 2.(b), we propose a novel HTG transformer framework (denoted as HTGT) to jointly integrate both spatial and temporal dependencies while preserving the heterogeneity to learn node representations over the HTG for malware detection. *To the best of our knowledge, this is* ***the first work*** *of devising a transformer framework over HTG to jointly characterize malware propagation and evolutionary patterns for their detection.* More specifically, our proposed HTGT consists of a heterogeneous spatial transformer (HST) and a temporal transformer (TT): (1) in the HST (Figure 2.(b.1)), we design heterogeneous node and relation encoders and derive different attentions over each node and edge to learn dedicated representations for different types of entities and relations; (2) in the TT (Figure 2.(b.2)), we attentively aggregate the historical dependencies of a given node into the HTGT; and (3) the two transformers (i.e., HST and TT) work collaboratively in an iterative manner to learn node representations in the HTG. Later, the node (i.e., app) representations learned from HTGT will be fed to a multi-layer perceptron (MLP) to train the classifier (Figure 2.(c)) for the detection of evolving Android malware. Based on the large-scale and real sample collections from Tencent Security Lab, we perform extensive experimental studies; and promising results with comparison to state-of-the-art baselines and popular security products demonstrate the performance of Dr.Droid in evolving Android malware detection. Our major contributions include:

- ***Novel model to jointly characterize malware propagation and evolution.*** To combat the increasingly sophisticated Android malware, we propose to consider app content to learn their evolution patterns and contextual information (i.e., the environment where apps dwell) to learn their propagation patterns. To meet this objective, it calls for novel model for abstraction. In this work, we present an innovative HTG for jointly modeling malware propagation and evolution patterns *at the first attempt.* Such a comprehensive description of Android apps within the ecosystem makes malware evasion much more difficult and costly.
- ***Innovative transformer framework over HTG to learn node presentations for malware detection.*** Based on the built HTG, by exploiting latest advances in machine learning, we propose an *end-to-end transformer framework over HTG* (i.e., HTGT) to integrate both spatial and temporal dependencies while retaining the heterogeneity to learn latent representations of Android apps that encode both their propagation and evolution patterns for the detection of malicious ones. In the HTGT, the designed heterogeneous spatial transformer (i.e., HST) and temporal transformer (i.e., TT) work in an iterative and collaborative way for node representation learning in HTG. As the proposed learning paradigm is a general framework for HTG represenation learning, in addition to malware detection, it can also be readily applied to various dynamic network mining tasks, such as node classification, clustering and similarity search.
- ***An intelligent system deployed in anti-malware industry.*** We develop a practical system Dr.Droid integrating our proposed method for evolving Android malware detection and provide a comprehensive experimental study based on the large-scale and real sample collections from Tencent Security Lab, which demonstrates the performance of our developed system. Dr.Droid has been deployed in anti-malware industry to protect millions of users worldwide against evolving malware attacks.

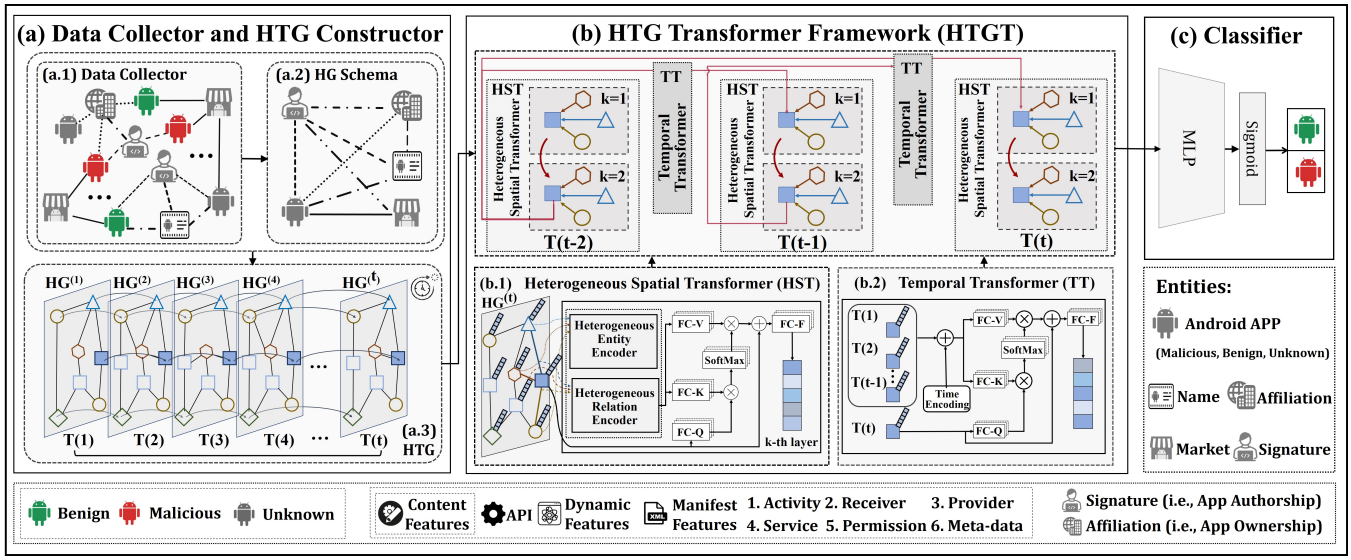


Figure 2: System architecture of Dr.Droid. In Dr.Droid, *(a)* based on the collected data, we first construct a HGT to jointly model Android app propagation and evolution for the abstraction; *(b)* to integrate both spatial and temporal dependencies while preserving heterogeneity of the built HGT, we then propose a novel HTGT framework, which consists of a heterogeneous spatial transformer (HST) and a temporal transformer (TT), to learn node representations in HTG in an iterative manner; and *(c)* the node representations learned from HTGT will be fed to a MLP to train the classifier for Android malware detection.

2 PROPOSED METHOD

In this section, we introduce our proposed method integrated in Dr.Droid for evolving Android malware detection in detail.

2.1 Feature Extraction

To comprehensively describe Android apps for evolving malware detection, we propose to consider content-based features of apps to learn their evolution and contextual information (i.e., the environment where apps dwell) to depict their propagation patterns.

Content Features of Android Apps. Android app is compiled and packaged in a single archive file (with an .apk suffix) that includes the source code in the dex file, resources, assets, and manifest file. As Application Programming Interfaces (APIs) are used by Android apps in order to access Android OS functionality and system resources, we extract the API calls from the dex file to describe a given app. For example, the set of APIs of ("*startActivity*", "*checkConnect*", "*sendSMS*", "*finishActivity*") extracted from a malicious trojan denote its intention of sending SMS messages without user's concern. Meanwhile, since the manifest file of an app describes essential information about the app to Android OS, we retrieve the activities, broadcast receivers, content providers, services, permissions and meta-data from the manifest file of each given app. For example, as activities provide Graphical User Interface (GUI) functionality to enable user interactivity, the extracted activities of "*com.assistant.home.LocationActivity*" and "*com.paypal.android.sdk.payments.LoginActivity*" from a COVID-19 themed malware reflect that it claims providing location-based COVID-19 updates but actually steals user's payment credential. In addition, we also obtain dynamic features during runtime executions of a given app when applicable, including its loaded dex files, connected urls and generated texts. For instance, during the runtime executions, an online banking trojan dynamically loaded six dex files from its C&C server to execute malicious activities and send user confidential to websites such as "*http://vpay.api.eeri*.com*" and "*http://pay.i*g.pw*". These statically and dynamically extracted features will be attached to each app as attributed feature vector.

Contextual Information. To capture the contextual information for the characterization of malware propagation patterns, we further extract higher-level semantic and social relations among apps and other entities within the ecosystem. The information include: ***R1***: *app-market* relation to denote an app and its hosted market (e.g., official app stores like Google Play, third-party markets such as Aptoide). For each market, we use one-hot representation as its attributed feature vector. ***R2***: *app-affiliation* relation to depict an app (e.g., "mobileqq") and its ownership (e.g., "tencent.com"). Companies conventionally use their reversed domain names (e.g., "tencent.com") to begin apps' package names (e.g., "com.tencent.mobileqq") which are unique names to identify the apps (e.g., "mobileqq"). ***R3***: *app-name* relation to describe an app and its abstraction. For each affiliation and name, we exploit pre-trained embedding model (e.g., Ernie [25]) to obtain its attributed features. ***R4***: *app-signature* relation to denote an app and its authorship (i.e., each app run on the Android must be signed by the developer, which relates to the app's signature). We also retrieve metadata such as the number/frequency a developer (i.e., app signature) posts/updates his/her apps in the markets as its attributed features. In addition, we further consider the *market-developer* relation (i.e., ***R5***) and interactions between app affiliations, signatures and names (i.e., ***R6-R8***).

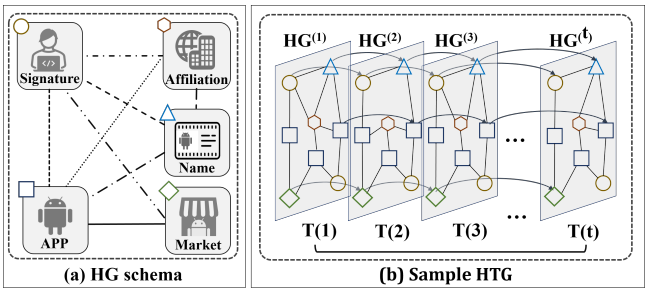
Given the rich semantic and complex relations extracted above, it is important to model them properly so that various relations can be well exploited to characterize app propagation, while considering the temporal dependencies to describe app evolution.

2.2 HTG Construction

To meet the above objective, we first present a structured heterogeneous graph (HG) that is capable of modeling heterogeneous spatial dependencies among different types of entities (i.e., nodes) to characterize app propagation patterns.

Definition 1. ***Heterogeneous Graph (HG).*** A HG is defined as a graph $G = (V, E, X)$ consisting of an entity set $V = \bigcup_{i=1}^{m} V_i$ of m type, a relation set E representing the spatial relations among entities, and a feature set $X = \bigcup_{i=1}^{m} X_i$ (X_i is the feature set for entity set V_i), with an entity type mapping $\phi: V \rightarrow L$ and a relation type mapping $\psi: E \rightarrow R$, where L denotes the entity type set and R is the relation type set, and $|L| + |R| > 2$. ***Graph Schema.*** The graph schema for G, denoted as $\mathcal{T}_G = (L, R)$, is a graph with nodes as entity types from L and edges as relation types from R.

HG not only provides the graph structure of data associations, but also provides a higher-level abstraction of the categorical association. In our application, we have five entity types (i.e., app, market, affiliation, name, signature) and eight types of relations among them (i.e., *R1-R8*). Based on the above definition, the HG schema is shown in Figure 3.(a). As such, a pattern of $app \xrightarrow{host^{-1}} market \xrightarrow{associate^{-1}} signature \xrightarrow{associate} market \xrightarrow{host} app$ can be encoded in the HG to characterize a malware propagation path - e.g., a developer distributed malicious apps through different markets.



(a) HG schema (b) Sample HTG

Figure 3: HG schema and a sample HTG.

Based on the built HG, we propose to further consider the temporal dependencies in the HGs (e.g., dynamics of evolving malware variants) to build a heterogeneous temporal graph (denoted as HTG), whose definition is given below.

Definition 2. ***Heterogeneous Temporal Graph (HTG).*** A HTG is a graph that is defined as $\mathcal{G} = \left(\{G^{(t)}\}_{t=1}^{T}, E'\right)$, where T is the number of timestamps in a given window, $G^{(t)}$ is a HG at timestamp t, E' describes the temporal relations between $G^{(t)}$ and $G^{(t+1)}$ $(1 \leq t < T)$.

Based on the above definition, a sample HTG is shown in Figure 3.(b). To this end, the problem of Android malware detection can be considered as node (i.e., app) classification in the HTG. To solve this problem, we first formulate the concept of HTG representation learning in the following.

Problem 1. ***HTG Representation Learning.*** Given a HTG $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, the representation learning task is to learn a function $\eta: \mathcal{V} \rightarrow \mathbb{R}^d$ that maps each node $v \in \mathcal{V}$ to a vector in a d-dimensional space $\mathbb{R}^d$ ($d \ll |\mathcal{V}|$), by integrating both spatial and temporal dependencies among them while preserving the heterogeneity.

The dynamics of HTG with evolution in time and space entailed the heterogeneity present new challenges for node representation learning over HTG for malware detection. HTG representation learning is an emerging research field, especially in the context of HTG with the heterogeneity developing a research frontier from currently considered homogeneous graph structures [8, 14, 20, 29, 30, 38]. In this paper, we propose a novel HTG transformer framework (denoted as HTGT) to integrate both spatial and temporal dependencies while retaining the heterogeneity to learn latent representations of Android apps (i.e., nodes in the HTG) that encode both their propagation and evolution patterns for the detection of malicious ones. We describe our proposed HTGT in detail in the following section.

2.3 HTGT for Node Representation Learning

It is not yet well understood how to integrate information in space and time into a single, general model for node presentation learning [19], especially in the context of heterogeneous graph structures. To tackle this challenge, as shown in Figure 2.(b), based on the constructed HTG, we propose a HTG transformer framework (i.e., HTGT) consisting of a heterogeneous spatial transformer (HST) and a temporal transformer (TT): in the HST, we design heterogeneous node and relation encoders and derive different attentions over each node and edge to learn dedicated representations for different types of entities and relations; in the TT, we attentively aggregate the historical dependencies of a given node into the HTGT. The two transformers of HST and TT are coupled and work collaboratively in an iterative manner to learn node representations in the HTG. More specifically, in each iteration: in the TT, each node receives information from its historical sequence; while in the HST, the nodes, which now have information from both present and past, start exchanging information through message passing. We elaborate our devised HST and TT in the followings.

Heterogeneous Spatial Transformer (HST). As shown in Figure 4, given a targeted node u^t and its neighborhood $\mathcal{N}_u^t$ in the HG at timestamp t (i.e., $G^{(t)} \in \mathcal{G}$), the representation of node u^t learned via HST can be formulated as:

$$\mathbf{z}_u^t = \text{HST}\Big(\mathbf{h}_u^t, \big\{(\mathbf{h}_v^t, e_{u,v}^t) : v \in \mathcal{N}_u^t\big\}; \Theta_{HST}\Big), \tag{1}$$

where $\mathbf{h}_v^t \in \mathbb{R}^d$ and $e_{u,v}^t$ are the d-dimension embedding of node v and the relation between u and its neighbor v, respectively, Θ_{HST} is the learnable parameters of HST and is shareable across all timestamps. To accommodate the node and relation heterogeneity, we equip HST with a node encoder $g(\cdot)$ and a relation encoder $f(\cdot)$. Formally, given u^t of type $\phi(u)$ and $e_{u,v}^t$ of type $\psi(u, v)$, $g(\cdot)$ and $f(\cdot)$ are defined as:

$$\begin{aligned} g(u) &= \mathbf{P}_{\phi(u)} \cdot \mathbf{h}_u^t, \\ f(u, v) &= \mathbf{R}_{\psi(u,v)} \cdot g(v), \end{aligned} \tag{2}$$

where $\mathbf{P}_{\phi(u)}, \mathbf{R}_{\psi(u,v)} \in \mathbb{R}^{d \times d}$ are node-dependent and relation-dependent weight matrices. The node encoder addresses the heterogeneity of a HG that originates from the node feature vectors while relation encoder deals with the heterogeneity that derives from different types of relations. Considering the fact that different neighbors have different impacts on the targeted node, inspired by the Transformer proposed in [26], we utilize its attention mechanism to implement HST by mapping the target node into a Query

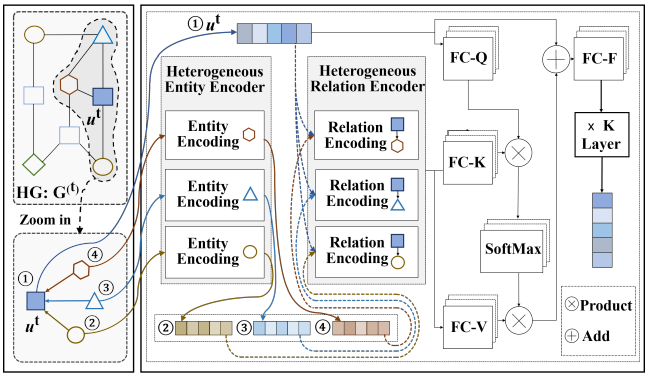


Figure 4: Heterogeneous spatial transformer (HST).

vector, neighbors into Key vectors and calculate their dot product as attentions. We define the $\mathbf{q}_u^t$ (query) for u, $\mathbf{k}_v^t$ (key) and $\mathbf{v}_v^t$ (value) for v in the heterogeneous spatial domain as:

$$\begin{aligned}\mathbf{q}_u^t &= \mathbf{W}_q^{HST} \cdot g(u),\\ \mathbf{k}_v^t &= \mathbf{W}_k^{HST} \cdot f(u,v),\\ \mathbf{v}_v^t &= \mathbf{W}_v^{HST} \cdot f(u,v),\end{aligned} \tag{3}$$

where $\mathbf{W}_q^{HST}, \mathbf{W}_k^{HST}, \mathbf{W}_v^{HST} \in \mathbb{R}^{d\times d}$ denote the transformation matrices for query, key, and value, respectively. $\mathbf{q}_u^t$ depicts the embedding of u that queries v, $\mathbf{k}_v^t$ represents the indexing of v and $\mathbf{v}_v^t$ depicts the transformed embedding for v. The attention α_v^t for v is calculated by:

$$\alpha_v^t = \frac{\exp(\mathbf{q}_u^t \cdot (\mathbf{k}_v^t)^\top)}{\sum_{v'\in\mathcal{N}_u^t} \exp(\mathbf{q}_u^t \cdot (\mathbf{k}_{v'}^t)^\top)}. \tag{4}$$

And then, the aggregated embedding $\mathbf{z}_u \in \mathbb{R}^d$ is formulated as:

$$\mathbf{z}_u^t = \mathbf{W}_f^{HST} \cdot \Big(g(u) + \sum_{v\in\mathcal{N}_u^t} \alpha_v^t \cdot \mathbf{v}_v^t\Big), \tag{5}$$

where $\mathbf{W}_f^{HST}$ is a regularization matrix which makes the aggregated embedding smooth. We employ multi-head attention to stabilize the learning process. Specifically, H independent attention heads are executed in a parallel fashion, and then their representations are averaged, resulting in the following output:

$$\mathbf{z}_u^t = \frac{1}{H}\sum_{h=1}^{H} \mathbf{W}_{f,h}^{HST} \cdot \Big(g(u) + \sum_{v\in\mathcal{N}_u^t} \alpha_{v,h}^t \cdot \mathbf{v}_{v,h}^t\Big). \tag{6}$$

Without loss of generality, we then stack K-layer HSTs in a back-to-back manner to capture K-order neighbor information.

Temporal Transformer (TT). To characterize app evolution, as shown in Figure 2.(b.2), our designed TT is equipped with an attention mechanism to capture the temporal dependencies. Formally, the representation of node u^t in the HG at timestamp t learned via TT is formulated as:

$$\begin{aligned}\mathbf{h}_u^t &= \mathrm{TT}(\mathbf{x}_u^t, Z_u^t; \Theta_{TT}),\\ Z_u^t &= \{\mathbf{z}_u^{t'} | t' < t\},\end{aligned} \tag{7}$$

where $\mathbf{x}_u^t \in \mathbb{R}^{d_{\phi(u)}}$ is the original feature vector of node u, Z_u^t denotes the historical sequences before timestamp t, Θ_{TT} is the learnable parameters of TT and is shareable across all timestamps. To capture the importance of a historical embedding $\mathbf{z}_u^{t'}$ with respect to the current feature vector $\mathbf{x}_u^t$, we transform $\mathbf{x}_u^t$ into Query vector, $\mathbf{z}_u^{t'}$ into Key vector and calculate their dot product as the attention. Motivated by [26], we further define a time encoding function $l(\cdot)$ for $\mathbf{z}_u^{t'}$ as:

$$\begin{aligned}l(\mathbf{z}_u^{t'}) &= \|_{i=1}^{d}\big(\mathbf{z}_u^{t'}(i) + p(t',i)\big),\\ p(t',i) &= \begin{cases}\sin(t'/10000^{2i/d}) & \text{if } i \text{ is even},\\ \cos(t'/10000^{2i/d}) & \text{if } i \text{ is odd},\end{cases}\end{aligned} \tag{8}$$

where i is the index of each element, $\|$ denotes the concatenation operation, $p(\cdot)$ is a frequency encoding function that characterizes a time-dependent sinusoid. The reason to incorporate $l(\cdot)$ is for TT to leverage time-related factors before calculating attentions for feature vectors at different timestamps. With this function, the feature vectors at different timestamps will be treated differently. Accordingly, $\mathbf{q}_u^t$, $\mathbf{k}_u^{t'}$ and $\mathbf{v}_u^{t'}$ in the temporal domain are formulated as:

$$\begin{aligned}\mathbf{q}_u^t &= \mathbf{W}_q^{TT} \cdot \mathbf{x}_u^t,\\ \mathbf{k}_u^{t'} &= \mathbf{W}_k^{TT} \cdot l(\mathbf{z}_u^{t'}),\\ \mathbf{v}_u^{t'} &= \mathbf{W}_v^{TT} \cdot l(\mathbf{z}_u^{t'}),\end{aligned} \tag{9}$$

where $\mathbf{W}_q^{TT} \in \mathbb{R}^{d\times d_{\phi(u)}}, \mathbf{W}_k^{TT}, \mathbf{W}_v^{TT} \in \mathbb{R}^{d\times d}$ denote the transformation matrices for query, key, and value, respectively. Finally, we apply Eq.(4) - Eq.(6) to conduct attention calculation, embedding aggregation and multi-head attention computation, respectively. To this end, our proposed HTGT to learn node representations in the HTG is given in Algorithm 1.

Algorithm 1: HTGT for Node Representation Learning

Input: HTG $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, time window T, HST layer K.
Output: Node representation $\mathbf{z}_u^T, \forall u \in \mathcal{V}$.

```
Initialize: Z ← {};
for t = 1 to T do
    q_u^t, k_u^{t'}, v_u^{t'} ← Eq.(8)-(9) with Z        t' < t, ∀u ∈ V^t;
    calculate temporal attention via Eq.(4);
    h_u^t ← Eq.(5)-(6)                                   ∀u ∈ V^t;
    for k = 1 to K do
        q_u^t, k_v^t, v_v^t ← Eq.(2)-(3) with h_u^t      ∀u ∈ V^t, ∀v ∈ N_u^t;
        calculate spatial attention via Eq.(4);
        z_u^t ← Eq.(5)-(6)                               ∀u ∈ V^t;
        h_u^t ← z_u^t                                    ∀u ∈ V^t;
    end
    append z_u^t to Z                                    ∀u ∈ V^t;
end
Return z_u^T                                             ∀u ∈ V;
```

2.4 Classification

By applying the proposed HTGT, given a node (i.e., app) u in the HTG, we are able to generate its representation $\mathbf{z}_u$. For malware detection task, as shown in Figure 2.(c), we then feed it into a MLP followed by a nonlinear layer (e.g., sigmoid) to get its prediction:

$$\hat{y}_u = \sigma(\mathrm{MLP}(\mathbf{z}_u)). \tag{10}$$

Given the training dataset $\mathcal{D}$, the HTGT can be back-propagated with the cross-entropy loss:

$$\mathcal{L} = -\frac{1}{|\mathcal{D}|} \sum_{u \in \mathcal{D}} y_u \log(\hat{y}_u) + (1 - y_u) \log(1 - \hat{y}_u) + \gamma \|\Theta\|_2^2, \quad (11)$$

where $\hat{y}_u$ and y_u are the prediction and ground-truth for node u respectively, and $\|\Theta\|_2^2$ is the L2-regularizer to prevent over-fitting. We also investigate the computational complexity of HTGT: given a HTG in time window T, for each iteration (i.e., $1 < t \leq T$), the complexity for TT to aggregate temporal information is $O(T|V^t|)$ and HST consumes $O(|E^t|)$ time for message passing; and thus the complexity of HTGT in each iteration is $O(T|V^t| + |E^t|)$ and $O(T|\mathcal{V}| + |\mathcal{E}|)$ for time window T.

3 EXPERIMENTAL RESULTS AND ANALYSIS

In this section, we conduct four sets of experiments using large-scale sample collections from Tencent Security Lab to fully evaluate the performance of Dr.Droid integrating our proposed method.

3.1 Experimental Setup

Data Collection and Preparation. We obtain the large-scale and real-world data collections from Tencent Security Lab: the sample set contains 82,831 Android apps collected from 55 app stores/ marketplaces/ websites during July 06 to July 12, 2020 (i.e., 15,041 of them are detected as malicious, 59,561 are benign, and 8,229 are unknown). For the 8,229 unknown apps, with the analysis by anti-malware experts in Tencent Security Lab, 3,096 are malicious and 5,133 are benign, which are used as the testing set in the experiments. After feature extraction, the constructed HTG has 210,947 nodes (i.e., 82,831 nodes with type of app, 44,816 nodes with type of signature (i.e., developer), 30,139 nodes with type of affiliation, 53,106 nodes with type of names, 55 nodes with type of market) and 1,531,887 edges (including 1,246,950 edges in heterogeneous spatial domain and 284,937 edges in temporal domain).

Decisions and tradeoffs for the system design choices. During the system development, we did encounter several challenges of finding optimal tradeoffs between detection performance and efficiency: how to decide the best time window while accommodating the scale of built HTG, and how to capture the temporal dependencies for the characterization of app evolution. To solve these problems, taking the domain knowledge from anti-malware experts, we set the time window to seven days as the life-cycle of active malware is about a week on average and we use a pre-trained framework to index the collected apps.

Environmental and Parameter Settings. Experiments are implemented under Ubuntu 19.10 operating system, equipped with Ryzen 9 3900X processor, two GeForce RTX 2080Ti graphic cards and 64 GB of RAM. Our model is developed under PyTorch 1.7.1 and Python 3.7.6. We utilize Adaptive Moment Estimation (Adam) to optimize HTGT with a learning rate of 0.005. We set the dimension of hidden embedding to 32, the number of epochs to 1000, the layers of HST to 2, each of which consists of 2 attention heads. We also use ReLU as the activation function and float32 operation to speed up training procedure and reduce memory usage. For reproducibility of experiments, we set the random seed of PyTorch and Numpy to 12,345. The open-source code of our proposed HTGT framework integrated in Dr.Droid is publicly available in GitHub [7].

Evaluation Metrics. To quantitatively assess performances of different methods in Android malware detection, we exploit the indices of true/false positives/negatives (TPs/FPs/TNs/FNs), precision, recall, accuracy (ACC) and F1 for evaluations.

3.2 Evaluation of Dr.Droid

In this set of experiments, we comprehensively validate the proposed methods integrated in Dr.Droid, including HTG modeling and HTGT representation learning framework in evolving Android malware detection. To validate the benefits of HTG for jointly modeling malware propagation and evolution by considering the heterogeneous spatial dependencies and temporal dimensions, we compare it with two types of app representations using the deep neural network (DNN) with a 3-layer MLP as classification model: $\text{DNN}_{content}$ taking the content features of apps as inputs, and $\text{DNN}_{augment}$ using the augmented features as inputs (i.e., content-based features concatenating relations of *R1-R4*). In order to examine how much performance each domain contributes to the detection, we also compare HST that is only with heterogeneous spatial dependencies to TT which only considers the temporal dimensions. In addition, we also validate the iterative working mechanism for HST and TT in our proposed HTGT framework by comparing it to two variants: (a) HTGT_{TS} that first uses sequential model to capture temporal dependencies across all timestamps and then performs message passing in heterogeneous spatial domain; and (b) HTGT_{ST} with the implementation in a reverse way.

Quantitative evaluation. The experimental results are shown in Table 1, from which we can see that: (1) with the help of additional relations, $\text{DNN}_{augment}$ performs better than $\text{DNN}_{content}$; (2) TT and HST achieve better results than traditional feature-based methods, which demonstrates the effectiveness of TT utilizing temporal dependence for depicting app evolution and HST integrating heterogeneous spatial dependencies to characterize app propagation for malware detection; (3) solely HST performs better than merely TT in our application, which indicates app propagation may provide higher-level generation than app evolution in malware detection; (4) Dr.Droid with the devised HTGT that jointly characterizes both app propagation and evolution patterns in an iterative manner outperforms others in detecting evolving Android malware, which achieves an impressive performance of 0.9880 ACC and 0.9908 F1.

Table 1: Evaluation of Dr.Droid in malware detection.

Model	ACC	F1	Recall	Precision
Detection based on traditional features				
$\text{DNN}_{content}$	0.9491	0.9253	0.9647	0.8889
$\text{DNN}_{augment}$	0.9539	0.9315	0.9604	0.9043
Contributions made by different domains				
TT	0.9643	0.9700	0.9673	0.9768
HST	0.9720	0.9775	0.9745	0.9819
HTGT with different settings				
HTGT_{TS}	0.9769	0.9812	0.9823	0.9816
HTGT_{ST}	0.9796	0.9834	0.9819	0.9846
Dr.Droid	**0.9880**	**0.9908**	**0.9913**	**0.9902**

Case studies and analysis for novel results. To better interpret the contributions made by different domains in evolving malware detection, we further perform the case studies based on the detection results. Figure 5.(a) shows that the evolving adware variants repackaged themselves on various online games to push unwanted apps and ads, whose latent malice (e.g., including malicious activities such as "*cn.**push.android.ui.PushActivity*" and "*com.b***e.sdk.openadsdk.activity.TTDelegateActivity*") can be derived by the designed TT. This demonstrates the benefits of TT which considers the temporal dependencies to capture malware evolution for their detection. By contrast, Figure 5.(b) shows an example that evolving scamware (e.g., "Earn Money Everyday") and malicious video players (e.g., "Sweet Orange") were disseminated via dedicated app supply chain; as such, Dr.Droid that integrates both heterogeneous spatial dependencies and temporal dimensions to capture malware propagation and evolution patterns shows its superiority in their detection.

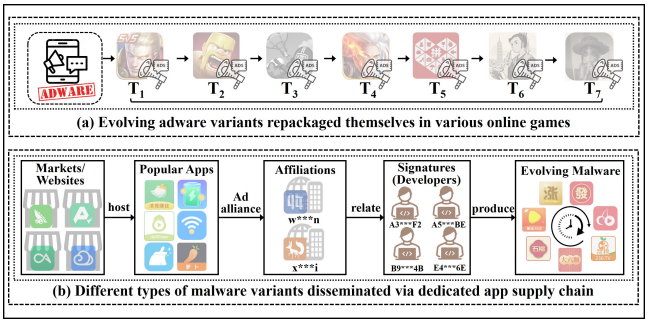


Figure 5: Case studies and analysis of novel results.

3.3 Comparison with Baseline Methods

In this set, we compare Dr.Droid incorporating the proposed HTGT for node representation learning in malware detection with state-of-the-art baseline methods. We consider three groups of baselines: homogeneous and heterogeneous graph representation learning models as well as dynamic HG learning methods.

- **Homogeneous graph embedding models:** Graph convolutional network (GCN) [17] averages neighbors' embeddings with a linear projection; graph attention network (GAT) [27] uses self-attention to aggregate information of neighbors.
- **Heterogeneous graph embedding methods:** Relational graph convolutional network (RGCN) [22] designs different linear projections for different types of relations for information aggregation; heterogeneous graph attention network (HAN) [28] aggregates the information from neighbors through node-level attention and semantic-level attention; heterogeneous graph transformer (HGT) [13] utilizes different transformer attention blocks for different relations, and summarizes embeddings from all blocks using a sum function.
- **Dynamic HG representation learning models:** We exploit our proposed temporal transformer (i.e., TT) to capture temporal dependencies while exploring HAN and HGT to characterize heterogeneous spatial dependencies for comparisons.

For GCN and GAT that are designed for homogeneous graphs, we simply ignore the heterogeneity and directly feed the graph into these learning models. For HAN, we devise four metapaths (i.e., *app-signature-app*, *app-affiliation-app*, *app-name-app* and *app-market-app*) to facilitate the node representation learning; in addition, we further project the distinct attribute dimensions of different node types into a unified feature space. The experimental results are shown in Table 2, from which we have the following observations: (1) heterogeneous graph embedding models (i.e., RGCN, HAN, HGT and our propose HST) outperform those homogeneous ones (i.e., GCN and GAT) as they are able to preserve richer spatial semantics and heterogeneous structure information, and note that the proposed HST performs better than the other three heterogeneous graph embedding models; (2) introducing temporal dependencies into malware detection task helps the performances (i.e., dynamic HG representation learning models obtain better results than others) as it helps capturing app evolution in addition to propagation patterns; (3) Dr.Droid that integrates HTGT for node representation learning over HTG achieves the best results in terms of all evaluation metrics. The improvement of HTGT against baselines is achieved by the well-designed HST that captures heterogeneity in spatial domain and the iterative learning mechanism that collaboratively couples HST and TT to jointly characterize app propagation and evolution for evolving malware detection.

Table 2: Comparison with baseline methods.

Method	ACC	F1	Recall	Precision
Homogeneous graph embedding methods				
GCN	0.9536	0.9650	0.9592	0.9724
GAT	0.9555	0.9682	0.9607	0.9754
Heterogeneous graph embedding models				
RGCN	0.9608	0.9686	0.9590	0.9730
HAN	0.9677	0.9748	0.9713	0.9822
HGT	0.9684	0.9757	0.9745	0.9791
HST	0.9720	0.9775	0.9745	0.9819
Dynamic HG representation learning methods				
TT+HAN	0.9829	0.9880	0.9868	0.9874
TT+HGT	0.9836	0.9881	0.9870	0.9887
Dr.Droid	**0.9880**	**0.9908**	**0.9913**	**0.9902**

3.4 Comparison with Other Detection Systems

In this set, we compare Dr.Droid with some popular commercial mobile security products and machine learning-based detection systems. For anti-malware products, we use Norton with the latest version of 5.1.0.5628 and Lookout with the latest version of 10.36.3-877b68b. For machine learning-based systems, we choose recently developed HG-based detection systems for comparison: HinDroid [12] that generates the commuting matrix for each metapath and uses multi-kernel learning to aggregate them for Android malware detection; AiDroid [35] that learns out-of-sample node representations for malware detection; and Dr.HIN [10] which explores disentangled representation learning in HG to learn distinct, informative factors hidden in HG embeddings for malware detection.

The results shown in Table 3 demonstrate that Dr.Droid outperforms the two anti-malware products and three HG-based learning

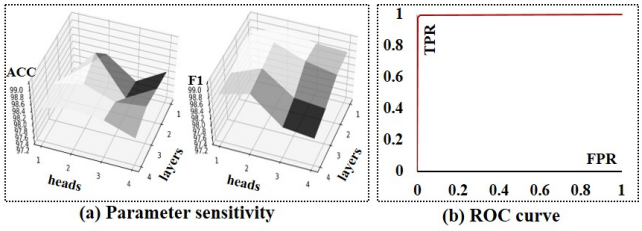
systems in detecting evolving Android malware. The main reasons behind this could be: (1) compared to the commercial security products, by bringing new innovations to jointly model heterogeneous spatial dependencies and temporal dimensions, Dr.Droid is able to characterize both malware propagation and evolution for their detection; (2) compared with the previously developed HG-based detection systems, Dr.Droid further introduces the temporal dependencies to depict malware evolution, which helps the detection.

Table 3: Comparison with other detection systems.

System	ACC	F1	Recall	Precision
Norton	0.9560	0.9622	0.9523	0.9734
Lookout	0.9559	0.9636	0.9583	0.9693
HinDroid	0.9605	0.9692	0.9634	0.9773
AiDroid	0.9653	0.9739	0.9658	0.9746
Dr.HIN	0.9704	0.9754	0.9730	0.9776
Dr.Droid	**0.9880**	**0.9908**	**0.9913**	**0.9902**

3.5 Evaluation of Parameter Sensitivity

In this set of experiments, we first investigate how different choices of hyper-parameters affect the model performance. In Dr.Droid, for the designed HTGT, there are two main parameters: the numbers of attention heads and layers of HST. We range them from one to four to examine their correlations with ACC and F1. For a better visualization, we plot two 3D graphs as shown in Figure 6.(a). We notice that both ACC and F1 achieve local optima with two spatial attention heads and layers of HST. By observing the gradient of these two surfaces, we find that increasing HST layers slightly helps boosting the performance; however, the performance would be deteriorated when considering a large number of layers as some unrelated neighbors could be brought. As such, Dr.Droid is able to reach high performance under a cost-effective parameter choice.



(a) Parameter sensitivity (b) ROC curve

Figure 6: Parameters and post-launch performance.

4 SYSTEM DEPLOYMENT AND IMPACTS

Deployment challenges and post-launch performance. As Android malware techniques are constantly evolving, the system has been updated on a daily basis to learn and incorporate newly detected malware against novel threats. Dr.Droid has been deployed and tested based on the large-scale and daily sample collections from Tencent Security Lab for over seven months: based on the newly collected apps everyday (i.e., about 150,000 testing samples per day), as shown in Figure 6.(b), the receiver operating characteristic (ROC) curve of Dr.Droid in detecting newly unknown Android malware achieves an impressive 0.9839 true positive rate (TPR) at 0.0043 false positive rate (FPR).

Novelty and usability. Thanks to the novel HTGT framework that is capable of jointly modeling malware propagation and evolution at the first attempt, the system has demonstrated its performance in evolving Android malware detection. Using the developed Dr.Droid, the analysis of daily collected unknown apps can be performed within hours with multiple servers.

Audience and societal impacts. The importance of cybersecurity can hardly be understated; by advancing data-driven innovations, Dr.Droid has been deployed in anti-malware industry to provide service for over 700 millions mobile users worldwide against evolving Android malware attacks.

5 RELATED WORK

In recent years, systems applying data mining and machine learning techniques have been developed for Android malware detection [3, 11, 16, 34, 37], where different kinds of classification models are constructed based on different feature representations. Different from most of the existing works that merely leveraged app content, in our prior works, HinDroid [12], AiDroid [35] and Dr.HIN [10] were proposed to exploit structured HGs to model the complex relations among apps and other types of entities for Android malware detection. These systems have been successfully deployed in anti-malware industry by tackling different challenges of representation learning over HGs: HinDroid [12] proposed multi-kernel learning to aggregate different similarities formulated by different meta-paths; AiDroid [35] resolved out-of-sample node representation learning problem; while Dr.HIN [10] explored disentangled representation learning in HG. Different from our preliminary studies, in this work, we exploit temporal dependencies of HGs to jointly model malware propagation and evolution for their detection at the first attempt.

Spatial-temporal graph representation learning has been widely studied [1, 20, 31, 39] with successful applications such as traffic flow prediction [4, 6, 8, 29, 30, 38] and infectious disease forecasting [15, 36]. However, most of the existing works mainly focus on homogeneous graph structures. To consider the spatial heterogeneity [24], dynamic HG representation learning has achieved increasing attention [9, 13, 18, 32, 33]. Those current works can be summarized in two categories: one first explores sequential models to capture temporal dependencies across all timestamps and then performs message passing in the spatial domain [5, 6, 15, 30, 38]; and the other first conducts message passing in each graph slice and then uses sequential models to obtain final representations [4, 8, 18, 29, 31, 32, 39]. It is not yet well understood how to integrate information in spatial and temporal domains into a single, general model for node presentation learning [19], especially in the context of heterogeneous graph structures. Different from existing studies, in this paper, leveraging advances of Transformer [26], we devise a novel framework consisting of a heterogeneous spatial transformer and a temporal transformer which work collaboratively in an iterative manner for HTG representation learning.

6 CONCLUSION

To protect legitimate users against evolving Android malware attacks, in this paper, we propose and develop an intelligent system named Dr.Droid to jointly model malware propagation and evolution for their detection *at the first attempt*. In Dr.Droid, we first

present a structured HG to model the complex semantic and social relations within the ecosystem to characterize app propagation patterns; to further depict app evolution, we also consider the temporal dependence and introduce a HTG to jointly model malware propagation and evolution by considering heterogeneous spatial dependencies with temporal dimensions. Based on the constructed HTG, we propose a novel HTG transformer framework (i.e., HTGT) to integrate both spatial and temporal dependencies while preserving the heterogeneity to learn node representations for malware detection. In our proposed HTGT, to preserve the heterogeneity, we devise a heterogeneous spatial transformer (i.e., HST) to derive heterogeneous attentions over each node and edge to learn dedicated representations for different types of entities and relations; to model temporal dependencies, we design a temporal transformer (i.e., TT) into the HTGT to attentively aggregate its historical sequences of a given node (i.e., app); the two transformers work collaboratively in an iterative fashion for representation learning. Comprehensive experiments based on the large-scale sample collections from Tencent Security Lab demonstrate the performance of Dr.Droid, which has been deployed in anti-malware industry to protect millions of users worldwide against evolving Android malware attacks.

7 ACKNOWLEDGMENTS

Y. Fan, M. Ju, S. Hou and Y. Ye's work is partially supported by the NSF under grants IIS-2027127, IIS-2040144, IIS-1951504, CNS-2034470, CNS-1940859, CNS-1814825, OAC-1940855 and ECCS-2026612, the DoJ/NIJ under grant NIJ 2018-75-CX-0032.

REFERENCES

[1] Emre Aksan, Peng Cao, Manuel Kaufmann, and Otmar Hilliges. 2020. A Spatio-temporal Transformer for 3D Human Motion Prediction. *arXiv preprint arXiv:2004.08692* (2020).
[2] AV-Test. 2021. *AV-Test Malware Statistics.* https://www.av-test.org/en/statistics/malware/.
[3] Haipeng Cai, Na Meng, Barbara Ryder, and Daphne Yao. 2019. Droidcat: Effective android malware detection and categorization via app-level profiling. *IEEE TIFS* 14, 6 (2019), 1455–1470.
[4] Zhiyong Cui, Kristian Henrickson, Ruimin Ke, and Yinhai Wang. 2019. Traffic graph convolutional recurrent neural network: A deep learning framework for network-scale traffic learning and forecasting. *IEEE TITS* 21, 11 (2019), 4883–4894.
[5] Songgaojun Deng, Shusen Wang, Huzefa Rangwala, Lijing Wang, and Yue Ning. 2019. Graph message passing with cross-location attentions for long-term ili prediction. *arXiv preprint arXiv:1912.10202* (2019).
[6] Zulong Diao, Xin Wang, Dafang Zhang, Yingru Liu, Kun Xie, and Shaoyao He. 2019. Dynamic spatial-temporal graph convolutional neural networks for traffic forecasting. In *AAAI*, Vol. 33. 890–897.
[7] Yujie Fan, Mingxuan Ju, Shifu Hou, Yanfang Ye, Wenqiang Wan, Kui Wang, Yinming Mei, and Qi Xiong. 2021. *Open-source Code of Dr.Droid.* https://github.com/kdd2021drdroid/KDD2021_DrDroid/tree/main.
[8] Shengnan Guo, Youfang Lin, Ning Feng, Chao Song, and Huaiyu Wan. 2019. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In *AAAI*, Vol. 33. 922–929.
[9] Huiting Hong, Yucheng Lin, Xiaoqing Yang, Zang Li, Kung Fu, Zheng Wang, Xiaohu Qie, and Jieping Ye. 2020. HetETA: Heterogeneous information network embedding for estimating time of arrival. In *KDD*. 2444–2454.
[10] Shifu Hou, Yujie Fan, Mingxuan Ju, Yanfang Ye, Wenqiang Wan, Kui Wan, Yinming Mei, Qi Xiong, and Fudong Shao. 2021. Disentangled Representation Learning in Heterogeneous Information Network for Large-scale Android Malware Detection in the COVID-19 Era and Beyond. In *AAAI*.
[11] Shifu Hou, Aaron Saas, Lifei Chen, and Yanfang Ye. 2016. Deep4maldroid: A deep learning framework for android malware detection based on linux kernel system call graphs. In *2016 IEEE/WIC/ACM International Conference on Web Intelligence Workshops (WIW)*. IEEE, 104–111.
[12] Shifu Hou, Yanfang Ye, Yangqiu Song, and Melih Abdulhayoglu. 2017. Hindroid: An intelligent android malware detection system based on structured heterogeneous information network. In *KDD*. ACM, 1507–1515.
[13] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. 2020. Heterogeneous graph transformer. In *WWW*. 2704–2710.
[14] Ziyu Jia, Youfang Lin, Jing Wang, Ronghao Zhou, Xiaojun Ning, Yuanlai He, and Yaoshuai Zhao. 2020. Graphsleepnet: Adaptive spatial-temporal graph convolutional networks for sleep stage classification. In *IJCAI*. 1324–1330.
[15] Amol Kapoor, Xue Ben, Luyang Liu, Bryan Perozzi, Matt Barnes, Martin Blais, and Shawn O'Banion. 2020. Examining covid-19 forecasting using spatio-temporal graph neural networks. *arXiv preprint arXiv:2007.03113* (2020).
[16] TaeGuen Kim, BooJoong Kang, Mina Rho, Sakir Sezer, and Eul Gyu Im. 2019. A Multimodal Deep Learning Method for Android Malware Detection Using Various Features. *IEEE TIFS* 14, 3 (2019), 773–788.
[17] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).
[18] Wenjuan Luo, Han Zhang, Xiaodi Yang, Lin Bo, Xiaoqing Yang, Zang Li, Xiaohu Qie, and Jieping Ye. 2020. Dynamic Heterogeneous Graph Neural Network for Real-time Event Prediction. In *KDD*. 3213–3223.
[19] Andrei Nicolicioiu, Iulia Duta, and Marius Leordeanu. 2019. Recurrent space-time graph neural networks. *arXiv preprint arXiv:1904.05582* (2019).
[20] Chiara Plizzari, Marco Cannici, and Matteo Matteucci. 2020. Spatial temporal transformer network for skeleton-based action recognition. *arXiv preprint arXiv:2008.07404* (2020).
[21] Purplesec. 2021. 2020 Ransomware Statistics, Data, and Trends. In *https://purplesec.us/resources/cyber-security-statistics/ransomware/.*
[22] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *ESWC*. Springer, 593–607.
[23] Statcounter. 2021. Mobile Operating System Market Share Worldwide. In *https://gs.statcounter.com/os-market-share/mobile/worldwide.*
[24] Yizhou Sun, Jiawei Han, Xifeng Yan, Philip S. Yu, and Tianyi Wu. 2011. PathSim: Meta Path-Based Top-K Similarity Search in Heterogeneous Information Networks. *PVLDB* (2011).
[25] Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Xuyi Chen, Han Zhang, Xin Tian, Danxiang Zhu, Hao Tian, and Hua Wu. 2019. Ernie: Enhanced representation through knowledge integration. *arXiv preprint arXiv:1904.09223* (2019).
[26] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762* (2017).
[27] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).
[28] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. 2019. Heterogeneous graph attention network. In *WWW*. 2022–2032.
[29] Xiaoyang Wang, Yao Ma, Yiqi Wang, Wei Jin, Xin Wang, Jiliang Tang, Caiyan Jia, and Jian Yu. 2020. Traffic flow prediction via spatial temporal graph neural network. In *WWW*. 1082–1092.
[30] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. 2019. Graph wavenet for deep spatial-temporal graph modeling. *arXiv preprint arXiv:1906.00121* (2019).
[31] Mingxing Xu, Wenrui Dai, Chunmiao Liu, Xing Gao, Weiyao Lin, Guo-Jun Qi, and Hongkai Xiong. 2020. Spatial-temporal transformer networks for traffic flow forecasting. *arXiv preprint arXiv:2001.02908* (2020).
[32] Hansheng Xue, Luwei Yang, Wen Jiang, Yi Wei, Yi Hu, and Yu Lin. 2020. Modeling Dynamic Heterogeneous Network for Link Prediction using Hierarchical Attention with Temporal RNN. *arXiv preprint arXiv:2004.01024* (2020).
[33] Luwei Yang, Zhibo Xiao, Wen Jiang, Yi Wei, Yi Hu, and Hao Wang. 2020. Dynamic heterogeneous graph embedding using hierarchical attentions. In *ECIR*. Springer, 425–432.
[34] Yanfang Ye, Lingwei Chen, Shifu Hou, William Hardy, and Xin Li. 2018. DeepAM: a heterogeneous deep learning framework for intelligent malware detection. *Knowledge and Information Systems* 54, 2 (2018), 265–285.
[35] Yanfang Ye, Shifu Hou, Lingwei Chen, Jingwei Lei, Wenqiang Wan, Jiabin Wang, Qi Xiong, and Fudong Shao. 2019. Out-of-sample Node Representation Learning for Heterogeneous Graph in Real-time Android Malware Detection. In *IJCAI*. 4150–4156.
[36] Yanfang Ye, Shifu Hou, Yujie Fan, Yiming Zhang, Yiyue Qian, Shiyu Sun, Qian Peng, Mingxuan Ju, Wei Song, and Kenneth Loparo. 2020. α-Satellite: An AI-Driven System and Benchmark Datasets for Dynamic COVID-19 Risk Assessment in the United States. *IEEE Journal of Biomedical and Health Informatics* 24, 10 (2020), 2755–2764.
[37] Yanfang Ye, Tao Li, Donald Adjeroh, and S Sitharama Iyengar. 2017. A survey on malware detection using data mining techniques. *ACM Computing Surveys (CSUR)* 50, 3 (2017), 1–40.
[38] Bing Yu, Haoteng Yin, and Zhanxing Zhu. 2017. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. *arXiv preprint arXiv:1709.04875* (2017).
[39] Cunjun Yu, Xiao Ma, Jiawei Ren, Haiyu Zhao, and Shuai Yi. 2020. Spatio-Temporal Graph Transformer Networks for Pedestrian Trajectory Prediction. In *ECCV*. Springer, 507–523.