Disentangled Representation Learning in Heterogeneous Information Network for Large-scale Android Malware Detection in the COVID-19 Era and Beyond

Shifu Hou[1], Yujie Fan[1], Mingxuan Ju[1], Yanfang Ye[1*], Wenqiang Wan[2], Kui Wang[2], Yinming Mei[2], Qi Xiong[2], Fudong Shao[2]

[1] Department of Computer and Data Sciences, Case Western Reserve University, OH, USA
[2] Tencent Security Lab, Tencent, Guangdong, China

Abstract

In the fight against the COVID-19 pandemic, many social activities have moved online; society's overwhelming reliance on the complex cyberspace makes its security more important than ever. In this paper, we propose and develop an intelligent system named Dr.HIN to protect users against the evolving Android malware attacks in the COVID-19 era and beyond. In Dr.HIN, besides app content, we propose to consider higher-level semantics and social relations among apps, developers and mobile devices to comprehensively depict Android apps; and then we introduce a structured heterogeneous information network (HIN) to model the complex relations and exploit meta-path guided strategy to learn node (i.e., app) representations from HIN. As the representations of malware could be highly entangled with benign apps in the complex ecosystem of development, it poses a new challenge of learning the latent explanatory factors hidden in the HIN embeddings to detect the evolving malware. To address this challenge, we propose to integrate domain priors generated from different views (i.e., app content, app authorship, app installation) to devise an adversarial disentangler to separate the distinct, informative factors of variations hidden in the HIN embeddings for large-scale Android malware detection. This is *the first attempt* of disentangled representation learning in HIN data. Promising experimental results based on real sample collections from security industry demonstrate the performance of Dr.HIN in evolving Android malware detection, by comparison with baselines and popular mobile security products.

Introduction

The deadly outbreak of coronavirus disease (COVID-19) has posed grand challenges to human society. In the fight against the global pandemic, many social activities have moved online. As mobile devices connected to the Internet have become increasingly ubiquitous, 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. However, due to its large market share and open source ecosystem of development, Android attracts not only 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 (e.g., stealing user credentials, pushing unwanted apps [Ye et al. 2017]). Driven by considerable profits, there has been explosive growth of Android malware - e.g., it's reported that there were over 1.89 million new Android malware which infected more than 38 million smart phones in the first half of 2019[1]. More specifically, utilizing both fear and financial incentives, cyber attackers are using COVID-19 or coronavirus as a lure all over the spectrum of sophistication to spread malware to gain profits from the global health crisis (e.g., CovidLock is an Android ransomware that portrays itself as a coronavirus tracker); according to a recent report[2], the pandemic has sparked 72% ransomware growth. To combat the exponential growth of sophisticated Android malware, it points to an imminent need for innovative detection techniques to protect legitimate users against the attacks in the COVID-19 era and beyond.

The most significant line to protect users against Android malware attacks is mobile security products (e.g., Norton and Tencent Mobile Security). However, in the never-ending arms race, attackers always devise new tactics to evade their detection. For example, as shown in Figure 1.(a), to bypass the content-based detection, a malicious banking trojan (i.e., *App-2*) could adopt repackaging techniques making its codes highly similar as a legitimate app that is developed to provide COVID-19 updates (i.e., *App-1*). To combat with our adversaries, instead of merely using content-based features for the detection, higher-level semantics and social relations among apps within the ecosystem could provide complementary knowledge: as shown in Figure 1.(b).right, if a developer always repackages various apps to produce customized malware, then the legitimacy of apps generated by this developer is questionable (e.g., *App-4* could possibly be malicious); as illustrated in Figure 1.(c).right, if a mobile device user is an online game user, then it may be easier to get infected by online game trojans (i.e., app co-existence may help determine whether a given one is malicious or not). *How to comprehensively leverage these information to build an intelligent system that is able to understand the milieu of given apps for the detection of malicious ones?*

*Corresponding Author: yanfang.ye@case.edu

[1] https://m.qq.com/security_lab/news_detail_517.html
[2] https://www.skyboxsecurity.com/trends-report/

To solve the above problem, there are two major challenges that need to be addressed. The first one is *how to devise an integral model capable of comprehensively depicting the complex semantics and social relations* among apps (app-content, app-authorship, app-device, etc). The second one is *how to derive disentangled representations that could separate the distinct, informative factors of variations in the observed environment of given apps*. Although an app's authorship may facilitate the adjudication of its legitimacy, as shown in Figure 1.(b), compared with *App-4*, the judgment of *App-3* may be more difficult as it's generated by a new developer whose information (i.e., portrait) has not yet well known. Similarly, as illustrated in Figure 1.(c), compared with *App-6*, the prediction of *App-5* via app co-existence is more challenging since many of its co-existing apps' labels are unknown. An intelligent system is expected to be able to leverage the comprehensive social relations (e.g., app authorship, app installation) while separating the distinct, informative factors of variations hidden in the complex ecosystem to facilitate the detection of evolving malware.

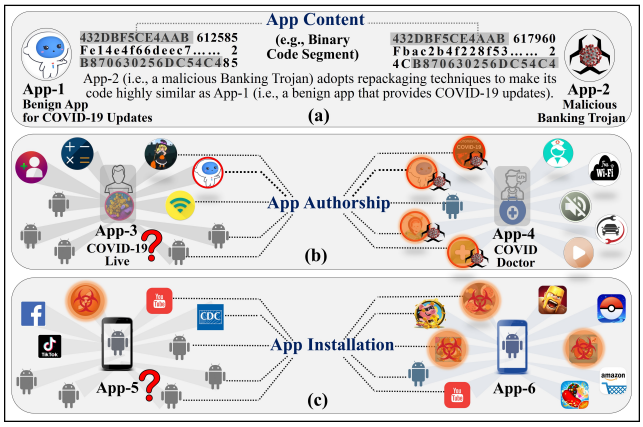


Figure 1: The complex and highly entangled ecosystem of Android apps for the detection of malicious ones.

In this paper, to address the above challenges, we propose and develop an intelligent system named Dr.HIN (shown in Figure 2) for large-scale Android malware detection to protect users against the attacks in the COVID-19 era and beyond. In Dr.HIN, to tackle the first challenge, as illustrated in Figure 2.(a), besides content-based features extracted from the given apps, we propose to consider higher-level semantics and social relations among apps, developers and devices to comprehensively depict Android apps; and then we introduce a structured heterogeneous information network (HIN) to model the complex relations and exploit meta-path guided strategy to learn node (i.e., app) representations from HIN. For the second challenge, an adversarial disentangler is expected to learn disentangled representations that can identify and disentangle the latent explanatory factors hidden in the observed data [Bengio, Courville, and Vincent 2013]. The disentangled representations have been demonstrated to be more robust, interpretable and controllable [Tran, Yin, and Liu 2017; Higgins et al. 2017; Dupont 2018], whose learning has gained considerable attention. Recently, disentangled representation learning has shown its success in the field of computer vision [Higgins et al. 2017; Dupont 2018]; however, the works on how to learn to disentangle the latent factors hidden in the network data have been scarce with focus on homogeneous networks without taking consideration of external knowledge (e.g., [Ma et al. 2019a,b; Liu et al. 2019; Wang et al. 2020; Hu et al. 2020; Guo et al. 2020]. **By far, there has no work on disentangled representation learning in HIN data.** To bridge this gap, in this work, we propose to integrate domain priors generated from different views (shown in Figure 2.(b)) to devise an adversarial disentangler (Figure 2.(c)) to separate the distinct, informative factors of variations in the HIN embeddings. Afterwards, the learned disentangled representations will be fed to deep neural network (DNN) to train the classifier (Figure 2.(d)) for the detection of Android malware. The major contributions of our work in this paper can be summarized as follows:

- *We comprehensively characterize Android apps from their higher-level semantics and social environment using the constructed HIN.* The comprehensive description modeled by HIN makes the evasion more difficult and costly.
- *We propose a novel disentangled representation learning framework in HIN at the first attempt.* As the representations of Android apps are highly entangled in the complex ecosystem of development (i.e., *technical assumption*), we propose an adversarial disentangler by integrating domain priors generated from three different views (i.e., app content, app authorship, app installation) to separate the distinct, informative factors of variations in the HIN embeddings. To the best of our knowledge, this is *the first work* of disentangled representation learning in HIN data (i.e., *theoretical contributions*). Though it's used for malware detection, the proposed learning paradigm is a general framework to identify and disentangle the latent explanatory factors hidden in HIN data and thus can be applied to various network mining tasks.
- *We develop an intelligent system that is deployed in anti-malware industry.* Comprehensive experimental studies and promising results based on the large-scale and real sample collections from industry demonstrate the performance of Dr.HIN, which has already been incorporated into the scanning tools of commercial mobile security products that protect millions of users worldwide.

Proposed Method

In this section, we introduce our proposed method of **d**isentangled **r**epresentation learning in **HIN** (i.e., **Dr.HIN**) for large-scale Android malware detection in detail.

Feature Extraction

To comprehensively describe Android apps for malware detection, besides app contents, we also consider higher-level semantics and social relations within the ecosystem.

Content-based Feature Extraction. 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) can be used by Android apps to access Android

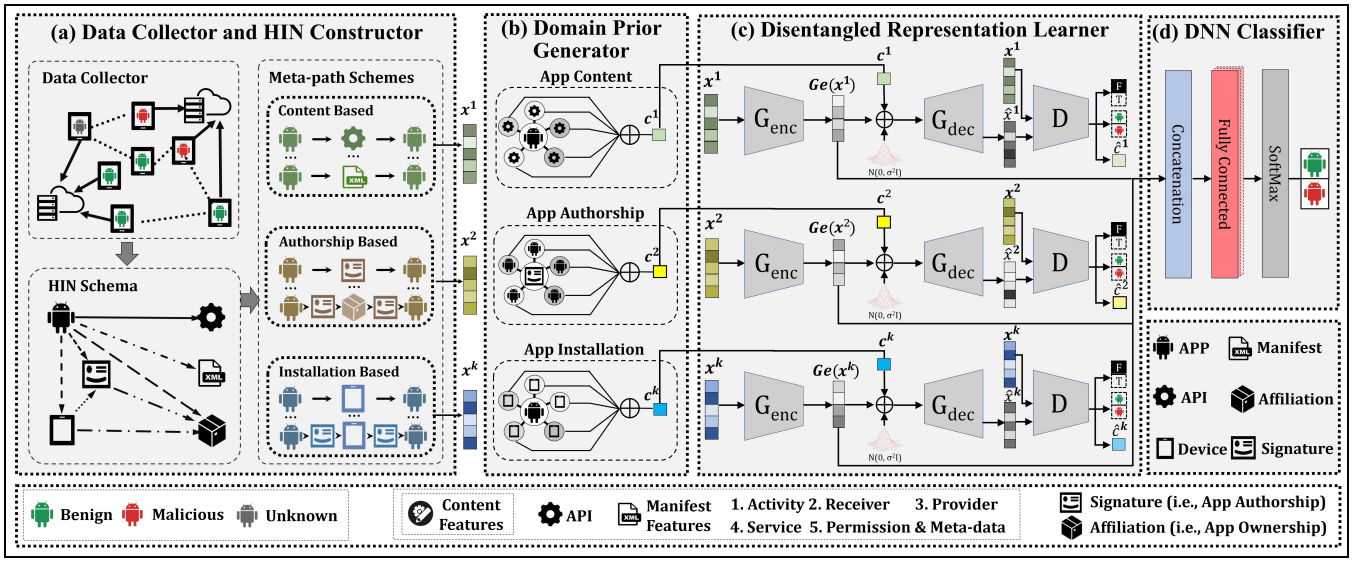


Figure 2: System architecture of Dr.HIN. In Dr.HIN, *(a)* we first construct a HIN to model different types of entities and relations and exploit meta-path guided strategy to learning node (i.e., app) representations in HIN (i.e., $\mathbf{x}$); and then *(b)* we propose to integrate domain priors (i.e., $\mathbf{c}$) generated from three views (i.e., app content, app authorship, app installation) to *(c)* devise an adversarial disentangler to separate the distinct, informative factors of variations in the HIN embeddings; afterwards, *(d)* the learned disentangled representations (i.e., $G_e(\mathbf{x})$) will be fed to DNN classifier for the detection of Android malware.

OS functionality and system resources, we extract the API calls from the dex file to describe a given Android app. For example, the set of APIs of {*"startActivity"*, *"checkConnect"*, *"sendSMS"*, *"finishActivity"*} extracted from a malicious "TigerEyeing" 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" reflect that a malicious banking trojan named "LocationAssistant" claims providing location-based COVID-19 updates but actually steals user's payment credential.

Relation-based Feature Extraction. To describe the relations between an app and its extracted content features, we use (1) *R1*: *app-invoke-API* to denote if an app invokes an API, and (2) *R2*: *app-include-manifest* to indicate if an app includes a specific activity, receiver, provider, service, permission or meta-data in the manifest file. We also extract (3) *R3*: *app-certify-signature* to denote an app and its authorship (i.e., each app running on the Android must be signed by the developer, which relates to the app's signature), and (4) *R4*: *app-associate-affiliation* to describe an app and its ownership: 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"). To detect the increasingly sophisticated Android malware, we further consider following social relations within the ecosystem: (5) *R5*: *app-install-device* depicts if an app is installed in a mobile device which can be denoted by its unique International Mobile Equipment Identity (IMEI) number; (6) *R6*: *device-have-signature* denotes the relation between a device and an app developer; (7) *R7*: *device-possess-affiliation* depicts the relation between a device and an app owner whose app is installed in this device; (8) *R8*: *signature-link-affiliation* denotes the relation between a developer and an app owner (e.g., a developer repackages an app that is originally produced by an official public health department).

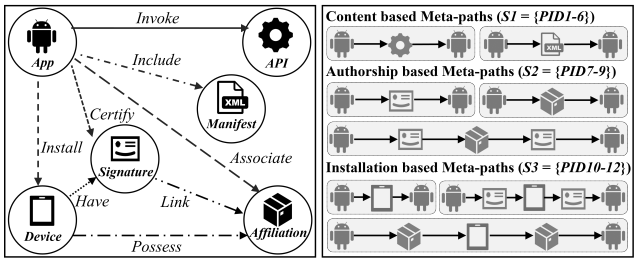
HIN Construction

Given the rich semantics and complex social relations extracted above, it is important to model them in a proper way so that different relations can be better and easier handled. We introduce HIN, which is able to be composed of multi-typed entities and relations, to solve this problem.

Definition 1 *A **heterogeneous information network (HIN)** [Sun et al. 2011] is defined as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with an entity type mapping ϕ: $\mathcal{V} \rightarrow \mathcal{A}$ and a relation type mapping ψ: $\mathcal{E} \rightarrow \mathcal{R}$, where $\mathcal{V}$ denotes the entity set and $\mathcal{E}$ is the relation set, $\mathcal{A}$ denotes the entity type set and $\mathcal{R}$ is the relation type set, and the number of entity types $|\mathcal{A}| > 1$ or the number of relation types $|\mathcal{R}| > 1$. The **network schema** for the $\mathcal{G}$, denoted as $\mathcal{T}_{\mathcal{G}} = (\mathcal{A}, \mathcal{R})$, is a graph with nodes as entity types from $\mathcal{A}$ and edges as relation types from $\mathcal{R}$.*

Based on the definitions above, the network schema in our application is shown in Figure 3.(a), where the apps are represented in a comprehensive way that utilizes both their semantic and structured information.

To formulate the relatedness among entities in HIN, we design three sets of meta-path schemes [Sun et al. 2011]: $S = \{S_i\}_{i=1}^{K}$ (K=3), where $S_1 = \{PIDj\}_{j=1}^{6}$, $S_2 = \{PIDj\}_{j=7}^{9}$, $S_3 = \{PIDj\}_{j=10}^{12}$, to characterize the relatedness over apps from three different views (i.e., app content, app authorship, app installation). For example, *PID1* depicts that two apps are related if they both invoke the same API, such as two malicious mobile video players both invoke the API of "*requestAudioFocus*"; *PID9* is able to describe that two apps (e.g., two banking trojans) produced by different developers both repackage same original app (e.g., a benign app that provides COVID-19 updates); and *PID10* denotes that two apps are associated if they co-exist in the same device.



(a) Network schema (b) Meta-paths

Figure 3: Network schema and designed meta-paths.

To this end, the problem of Android malware detection can be considered as node (i.e., app) classification in HIN. To solve this problem, we first present the concept of ***HIN representation learning*** [Fu, Lee, and Lei 2017]: given a HIN $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, the representation learning task is to learn a function $g : \mathcal{V} \rightarrow \mathbb{R}^d$ that maps each node $v \in \mathcal{V}$ to a vector in a d-dimensional space $\mathbb{R}^d$, $d \ll |\mathcal{V}|$ that are able to preserve the structured and semantic relations among them. To learn node representations in HIN, various embedding methods [Fu, Lee, and Lei 2017; Dong, Chawla, and Swami 2017; Fan et al. 2018] have been proposed. In this work, for each designed meta-path in $S_i \in S$, we exploit meta-path guided strategy [Ye et al. 2019] to learn node (i.e., app) representations in HIN, denoted as $\boldsymbol{X} = \{X^k\}_{k=1}^{K}$ (K=3).

Domain Prior Generation

Instead of using content-based features only, the constructed HIN provides a comprehensive way to depict the higher-level semantics and complex social relations among apps. However, detecting malware from the large-scale Android apps is a challenging task, as they are developed with complex interactions of many latent factors in the ecosystem (e.g., it's complicated that how apps are written, how developers reuse other apps to generate customized apps, how a set of apps are installed in user devices). For example, as discussed in Section 1, the distributions of apps in different devices (shown in Figure 1.(c)) may contribute differently to the determination of given apps' legitimacy. As external knowledge could make the learned disentangled representations more reliable [Locatello et al. 2019], *how to generate and incorporate domain priors to derive the distinct, informative factors hidden in the HIN embeddings* $\boldsymbol{X}$*?*

To solve this problem, we propose to generate the domain priors from three views (i.e., *app content*, *app authorship*, and *app installation*) to guide the learner to disentangle the latent explanatory factors hidden in the HIN embeddings. We here use app installation to illustrate how we generate the related domain priors. Suppose that there are N apps installed in T devices and each app is with label $y \in \{0, 0.5, 1\}$ (i.e., 0: benign, 1: malicious, 0.5: unknown), then the devices can be represented as $\mathbf{M} = [m_{ij}] \in \mathbb{R}^{T \times N}$, $m_{ij} \in \{1, 0\}$ (i.e., 1: if app j is installed in device i, 0: otherwise). For device i, we calculate its residual information:

$$Ires_{m_i} = \frac{\sum_{j=1}^{N}(m_{ij} = 1 \wedge y_j = 0.5)}{\sum_{j=1}^{N} m_{ij}}. \quad (1)$$

To this end, the prior of app j (denoted as c_j) can be obtained by average pooling of the residual information of all the devices where it installs:

$$c_j = \frac{\sum_{i=1}^{T}(Ires_{m_i} \times m_{ij})}{\sum_{i=1}^{T} m_{ij}}. \quad (2)$$

The generated prior c_j would guide the learner to disentangle the distinct factors for determining app j's legitimacy. Similarly, we can gain the domain priors from the perspectives of app content and app authorship. Then, for the HIN embeddings $\boldsymbol{X}$, we will have the corresponding domain priors, denoted as $\boldsymbol{C} = \{C^k\}_{k=1}^{K}$ (K=3).

Adversarial Disentangler in HIN

To incorporate the above generated domain priors $\boldsymbol{C}$ to derive the distinct, informative factors hidden in the HIN embeddings $\boldsymbol{X}$, we propose an adversarial disentangler (shown in Figure 4) to achieve this goal. The proposed adversarial disentangler consists of a generator G and a discriminator D: given a node (i.e., app) representation in HIN (i.e., $\mathbf{x}$), G aims to incorporate the related domain prior c and latent variables $\mathbf{z}$ (i.e., $\mathbf{z} \sim \mathcal{N}(0, \sigma)$) to produce a synthetic embedding $\hat{\mathbf{x}}$ via an encoder-decoder framework; D competes with G while assuring the detection performance and retaining the prior; and the disentangled representation $G_e(\mathbf{x})$ will be derived via the adversarial minimax game [Goodfellow et al. 2014]. We introduce the framework in detail below.

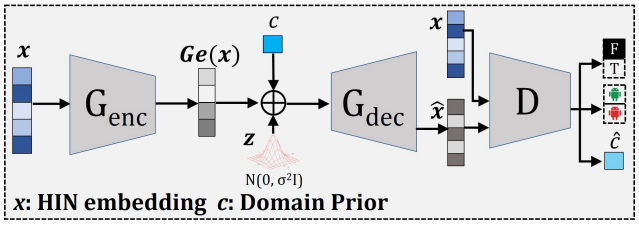


Figure 4: The framework of adversarial disentangler.

Multi-task Discriminator. The discriminator in the designed framework is a multi-task DNN consisting of three parts: $D = [D^T, D^L, D^C]$, where D^T aims to distinguish a real input from a synthetic one, D^L is to predict the class

label of the input (i.e., malicious or benign), and D^C is to decode the related domain prior (i.e., $c \sim p(c)$). Given an embedding input (i.e., either an original HIN embedding $\mathbf{x}$ or a synthetic embedding $\hat{\mathbf{x}} = G(\mathbf{x}, c, \mathbf{z})$), D aims to classify it as real or synthetic while estimating its class label and decoding the related domain prior. The objective function of D can be formulated as:

$$\max_D J_D(\Theta_D, \Theta_G) = \mathbb{E}_{p_{\text{data}}(\mathbf{x})}[\log D^T(\mathbf{x}) + \log D^L(\mathbf{x}) - \log D^C(\mathbf{x})] + \mathbb{E}_{p_{\text{model}}(\mathbf{x})}[\log(1 - D^T(\hat{\mathbf{x}})) + \log(D^L(\hat{\mathbf{x}})) - \log(D^C(\hat{\mathbf{x}}))]. \quad (3)$$

The first part of Eq. (3) is to maximize the probabilities of real input $\mathbf{x}$ being identified as true and correctly classified while retaining the domain prior (i.e., we use the cross-entropy loss for D^T and D^L, and mean squared error for D^C); the second part conducts similar computations for class label and domain prior but attempts to maximize the probability of synthetic $\hat{\mathbf{x}}$ being identified as false.

Disentangled Generator. Given a HIN embedding $\mathbf{x}$, through an encoder-decoder framework, the generator G aims to generate a synthetic embedding $\hat{\mathbf{x}}$ with the incorporation of the related domain prior c (i.e., app content, app authorship, and app installation). The generator is designed in the form of $G = [G_e, G_d]$, where the encoder G_e aims to learn a mapping from the original HIN embedding $\mathbf{x}$ to a disentangled representation $G_e(\mathbf{x})$, while the decoder G_d takes $G_e(\mathbf{x})$ associated with the domain prior c and latent variables $\mathbf{z}$ to produce $\hat{\mathbf{x}} = G_d(G_e(\mathbf{x}), c, \mathbf{z})$ aiming to fool D. The objective function of G can be formulated as:

$$\max_G J_G(\Theta_D, \Theta_G) = \mathbb{E}_{\substack{p_{\text{data}}(\mathbf{x}) \\ p(c), p(\mathbf{z})}}[\log(D^T(G_d(G_e(\mathbf{x}), c, \mathbf{z}))) + \log(D^L(G_d(G_e(\mathbf{x}), c, \mathbf{z}))) - \log(D^C(G_d(G_e(\mathbf{x}), c, \mathbf{z})))]. \quad (4)$$

Algorithm 1 illustrates our adversarial disentangler.

Classifier for Malware Detection

To this end, for a node (i.e., app) with its HIN embeddings $\{\mathbf{x}^k\}_{k=1}^K$ (K=3), their disentangled representations $\{G_e(\mathbf{x}^k)\}_{k=1}^K$ (K=3) will be derived by the above proposed adversarial disentangler. We then concatenate the disentangled representations as an input to train a DNN framework (as shown in Figure 2.(d)) for the classification task (i.e., detecting whether a given app is malicious or not). For the complexity analysis, given N apps, the HIN representation learning first consumes $O(N \log N)$ time to generate $N \times d$-dimensional embeddings as the inputs for the proposed adversarial disentangler; while the time complexity of the adversarial disentangler is $O(Nd^2K\prod h) \approx O(N)$, as K is the number of types of domain priors and h is the number of hidden neurons which are constant.

Experimental Results and Analysis

In this section, we conduct four sets of experiments using large-scale and real sample collections from anti-malware industry to fully evaluate the performance of Dr.HIN.

Algorithm 1: Adversarial Disentangler in Dr.HIN

Input: HIN node embeddings $\mathbf{X}$, domain priors $\mathbf{C}$, latent variables $\mathbf{z}$, node class labels L
Output: disentangled representation $G_e(\mathbf{x})$, $\mathbf{x} \in \mathbf{X}$

```
for each epoch do
    for each batch do
        Sample a half batch of real inputs < x, c, l >,
          where x ∈ X, c ∈ C, l ∈ L;
        Update Θ_D by Eq. (3);
        Generate related half batch of synthetic
          inputs < G_d(G_e(x), c, z), c, l >;
        Update Θ_D by Eq. (3);
        Freeze Θ_D;
        Generate a batch of synthetic inputs
          < G_d(G_e(x), c, z), c, l >;
        Update Θ_G by Eq. (4);
    end
end
return G_e(x)
```

Experimental Setup

Data Collection and Preparation. By collaboration with anti-malware industry, we obtain 75,397 apps queried by 291,822 mobile devices during the first week of July, 2020. In this large sample set, 25,179 apps are labeled as malicious and 39,057 are benign (i.e., training set total of 64,236 apps); for the remaining apps, we ask anti-malware experts in Tencent Security Lab to further analyze - i.e., 2,673 are malicious and 8,488 are benign (i.e., testing set total of 11,161 apps). After feature extraction and based on the network schema, the constructed HIN has 766,976 nodes and 249,610,168 edges. Table 1 shows the details of our data.

		Malware	Benign	Total		
APP	**Train**	25,179	39,057	**64,236**		
	Test	2,673	8,488	**11,161**		
	Entity	APP	Device	Entity	Sign	Affi
	Num	75,397	291,822	Num	5,611	7,558
HIN	Entity	API	Manifest	**Nodes**	**766,976**	
	Num	4,987	381,601	**Edges**	**249,610,168**	

Table 1: Details of collected data and built HIN.

Implementation. The experiments are conducted under the environment of Ubuntu 19.10, plus two Intel i9-9900k, 4-way SLI GeForce RTX 2080 Ti Graphics Cards and 64 GB of RAM, with the framework of Pytorch 1.3.1 and Python 3.7. For HIN embedding, we set the number of walks per node to 30, the walk length to 50, the window size to 5 and the dimension of node embedding d to 200; for disentangled representation learning, we set the dimension of disentangled embedding to 198, and perform Adam for optimization with learning rate of 0.0002.

Evaluation Metrics. We exploit *precision*, *recall*, *accuracy (ACC)* and *F1* as the metrics for performance evaluation.

Evaluation of Dr.HIN

We first comprehensively validate our proposed methods integrated in Dr.HIN, including HIN representations and the adversarial disentangler for Android malware detection. The results on testing set are shown in Table 2.

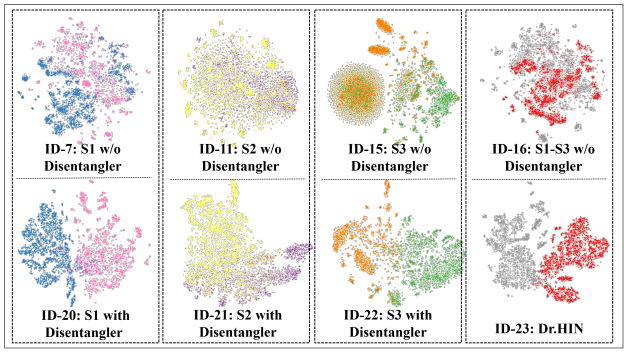
ID	Setting	Prior	*ACC*	*F1*
Comparisons of different meta-paths				
1	PID1	—	0.9568	0.8625
2	PID2	—	0.9720	0.9492
3	PID3	—	0.9500	0.8809
4	PID4	—	0.9434	0.8880
5	PID5	—	0.8919	0.8344
6	PID6	—	0.9586	0.9150
7	S_1={PID1-6}	—	0.9753	0.9496
8	PID7	—	0.9231	0.8176
9	PID8	—	0.9182	0.8051
10	PID9	—	0.9665	0.9325
11	S_2={PID7-9}	—	0.9707	0.9409
12	PID10	—	0.9374	0.6967
13	PID11	—	0.9518	0.8971
14	PID12	—	0.9546	0.8728
15	S_3={PID10-12}	—	0.9597	0.9168
HIN representations vs. traditional features				
16	f_1: {S_1-S_3}	—	0.9798	0.9579
17	f_2: Content	—	0.9751	0.9496
18	f_3: Relation	—	0.9145	0.8181
19	f_4: Augment	—	0.9755	0.9505
Evaluation of adversarial disentangler				
20	S_1={PID1-6}	C^1	0.9846	0.9676
21	S_2={PID7-9}	C^2	0.9731	0.9458
22	S_3={PID10-12}	C^3	0.9639	0.9220
23	**Dr.HIN**		**0.9896**	**0.9782**

Table 2: Evaluation of Dr.HIN in malware detection.

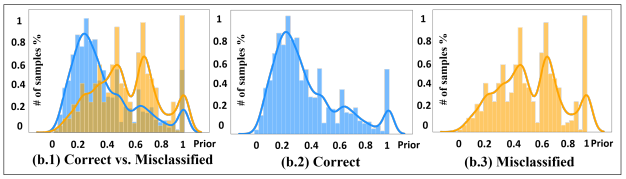
A. Evaluation of HIN Representations. From Table 2, we can see that: (1) Compared with single meta-path in each set, the combination of related meta-paths improves the performance. (2) The set of meta-paths depicting the relatedness over apps in terms of content-based correlations (i.e., ID-7) performs better than authorship-based (i.e., ID-11) and installation-based (i.e., ID-15) meta-paths. (3) Using the same DNN classifier, we compare HIN representations (i.e., combination of S_1-S_3, denoted as f_1) with three types of features: content-based features (i.e., APIs and manifest features, denoted as f_2), relation-based features (i.e., *R3-R5*, denoted as f_3) and augmented features (i.e., concatenation of f_2 and f_3, denoted as f_4). The results show that HIN representations (i.e., ID-16) encoding higher-level semantics are more expressive than the others (i.e., ID-17 to ID-19).

B. Evaluation of Adversarial Disentangler. In this set, we further examine the effectiveness of our proposed adversarial disentangler in HIN. The results in Table 2 demonstrate that: (1) Applying the adversarial disentangler with integration of domain priors indeed improves the detection performance (i.e., ID-20 vs. ID-7, ID-21 vs. ID-11, ID-22 vs. ID-15, ID-23 vs. ID-16). (2) Our proposed Dr.HIN achieves an impressive ACC of 0.9896 and F1 of 0.9782 in detecting evolving Android malware.

C. Visualization and Analysis for Novel Results. For a more intuitive comparison, we visualize the node representations before and after disentanglement via t-SNE [Maaten and Hinton 2008]. We plot the results in Figure 5.(a), which shows the benefits of disentangled representations in classification. To investigate the reasons behind, we use the generated HIN embeddings guided by installation-based meta-paths (i.e., ID-15) for further analysis: as shown Figure 5.(b), the correctly classified and misclassified apps are differently distributed under condition of domain priors (i.e., compared with correctly classified apps, the misclassified apps are with higher values of priors). Such *proof sketches/intuitions for novel results* indicate that the distinct factors of HIN embeddings for misclassified apps may be disentangled by integration of generated priors, which will thus help the detection.



(a) Visualization of representations w/o and with disentangler.



(b) Distributions of correctly classified and misclassified apps under condition of the priors (i.e., proof sketches/intuitions for novel results).

Figure 5: Visualization of representations and analysis.

Comparisons with Baseline Methods

In this section, we compare Dr.HIN with the state of the arts: i) homogeneous network embedding models (i.e., DeepWalk [Perozzi, Al-Rfou, and Skiena 2014], LINE [Tang et al. 2015]); and ii) HIN embedding methods (i.e., HIN2Vec [Fu, Lee, and Lei 2017], metapath2Vec [Dong, Chawla, and Swami 2017]). We also prepare two variants of Dr.HIN: one is to remove the priors (denoted as Dr.HIN$_{w/o}$); the other is with replacement of random noises (denoted as Dr.HIN$_{rnd}$).

The comparison results are shown in Table 3, which show: (1) HIN embedding methods (i.e., HIN2Vec, metapath2vec) achieve better performances than homogeneous network embedding models (i.e., DeepWalk, LINE). (2) The adversarial disentangler with our designed priors performs better than random condition and w/o priors. (3) Dr.HIN consistently outperforms all baselines for Android malware detection. The reasons behind this are i) the proper consideration and accommodation of the heterogeneity of HIN, and ii) the advantage of adversarial disentangler integrating domain priors that is able to learn the distinct and informative factors hidden in HIN data for malware detection.

Method	*ACC*	*F1*	*Recall*	*Precision*
DeepWalk	0.9683	0.9354	0.9596	0.9125
LINE	0.9298	0.8497	0.8290	0.8714
HIN2Vec	0.9779	0.9545	0.9670	0.9423
metapath2vec	0.9798	0.9579	0.9615	0.9543
Dr.HIN$_{w/o}$	0.9823	0.9623	0.9465	0.9787
Dr.HIN$_{rnd}$	0.9840	0.9666	0.9678	0.9653
Dr.HIN	**0.9896**	**0.9782**	**0.9731**	**0.9834**

Table 3: Comparison with baseline methods.

Comparisons with other Detection Systems

We also evaluate Dr.HIN by comparing with some popular commercial mobile security products (i.e., Norton and Lookout) and machine learning-based detection systems (i.e., HinDroid [Hou et al. 2017], Scorpion [Fan et al. 2018] and AiDroid [Ye et al. 2019]). The results shown in Table 4 demonstrate that Dr.HIN outperforms the two anti-malware products and three HIN-based learning systems in detecting evolving Android malware. Its success lies in the structured HIN for app representations and the adversarial disentangler for learning distinct, informative factors hidden in HIN.

System	*ACC*	*F1*	*Recall*	*Precision*
Norton	0.9560	0.9047	0.8709	0.9413
Lookout	0.9533	0.8986	0.8641	0.9359
HinDroid	0.9662	0.9276	0.9046	0.9519
Scorpin	0.9761	0.9500	0.9499	0.9502
AiDroid	0.9765	0.9510	0.9514	0.9507
Dr.HIN	**0.9896**	**0.9782**	**0.9731**	**0.9834**

Table 4: Comparison with other detection systems.

Evaluation of Model Stability

In this set, we examine the model stability of Dr.HIN: Figure 6.(a) plots the training losses of generator and discriminator in Dr.HIN, which demonstrates its training stability; Figure 6.(b) shows the ROC (receiver operating characteristic) curve of Dr.HIN, which achieves an impressive 0.9731 true positive rate (TPR) at 0.0052 false positive rate (FPR). Note that the computational complexity (i.e., scalability) analysis is given in the methodology section.

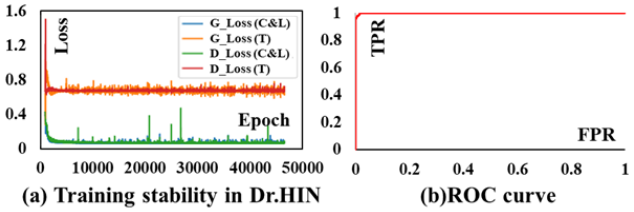


Figure 6: Evaluation of model stability.

Related Work

In recent years, systems applying machine learning techniques have been developed for malware detection (e.g., [Cai et al. 2018; Kim et al. 2018; Ye et al. 2017; Hou et al. 2016; Ye et al. 2008]). In particular, HIN-based models (i.e., HinDroid [Hou et al. 2017], Scorpion [Fan et al. 2018] and AiDroid [Ye et al. 2019]) have demonstrated the success in malware detection by tackling different challenges of HIN representation learning. The evolving Android malware (especially the ones using COVID-19 as lure to perform various malicious activities) have posed a new challenge of exploiting HIN-based models for the detection: HIN representations could be highly entangled within the complex ecosystem of app development, which calls for novel techniques to identify and disentangle the distinct, informative factors hidden in the HIN data for the detection of increasingly sophisticated malware. Although there have been many efforts on disentangled representation learning in computer vision [Tran, Yin, and Liu 2017; Higgins et al. 2017; Dupont 2018], disentangled representation learning in network data is an emerging field with main focus on homogeneous networks (e.g., [Ma et al. 2019a,b; Liu et al. 2019; Wang et al. 2020; Hu et al. 2020; Guo et al. 2020; Liu et al. 2020]. By far, there has no work on disentangled representation learning in HIN data. In this paper, we propose to integrate domain priors to devise an adversarial disentangler to bridge this gap and apply it for Android malware detection.

Conclusion

To combat the evolving Android malware, it calls for innovative detection techniques to protect users against the attacks in the COVID-19 era and beyond. To solve this problem, in this paper, besides app content, we propose to consider higher-level semantics and social relations among apps; and then we introduce a structured HIN to model the complex relations and exploit meta-path guided strategy to learn node (i.e., app) representations from HIN. Based on the HIN embeddings, we propose to integrate domain priors generated from different views to devise an adversarial disentangler at the first attempt to separate the distinct, informative factors of variations in HIN data for evolving Android malware detection. Comprehensive experimental studies and promising results based on the large-scale sample collections from anti-malware industry demonstrate that the developed system Dr.HIN incorporating our proposed method outperforms the state-of-the-arts baselines and popular commercial mobile security products in Android malware detection.

Acknowledgments

S. Hou, Y. Fan, M. Ju and Y. Ye's work is partially supported by the NSF under grants IIS-2027127, IIS-2040144, IIS-1951504, CNS-2034470, CNS-1940859, CNS-1946327, CNS-1814825, OAC-1940855 and ECCS-2026612, the DoJ/NIJ under grant NIJ 2018-75-CX-0032.

References

Bengio, Y.; Courville, A.; and Vincent, P. 2013. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35(8): 1798–1828.

Cai, H.; Meng, N.; Ryder, B.; and Yao, D. 2018. Droidcat: Effective android malware detection and categorization via app-level profiling. *IEEE Transactions on Information Forensics and Security* 14(6): 1455–1470.

Dong, Y.; Chawla, N. V.; and Swami, A. 2017. metapath2vec: Scalable representation learning for heterogeneous networks. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 135–144.

Dupont, E. 2018. Learning disentangled joint continuous and discrete representations. *arXiv preprint arXiv:1804.00104* .

Fan, Y.; Hou, S.; Zhang, Y.; Ye, Y.; and Abdulhayoglu, M. 2018. Gotcha-sly malware! Scorpion: a metagraph2vec based malware detection system. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 253–262.

Fu, T.-y.; Lee, W.-C.; and Lei, Z. 2017. Hin2vec: Explore meta-paths in heterogeneous information networks for representation learning. In *ACM International Conference on Information and Knowledge Management*, 1797–1806.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. In *NIPS*, 2672–2680.

Guo, X.; Zhao, L.; Qin, Z.; Wu, L.; Shehu, A.; and Ye, Y. 2020. Interpretable Deep Graph Generation with Node-Edge Co-Disentanglement. *arXiv preprint arXiv:2006.05385* .

Higgins, I.; Matthey, L.; Pal, A.; Burgess, C.; Glorot, X.; Botvinick, M.; Mohamed, S.; and Lerchner, A. 2017. beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework. *ICLR* 2(5): 6.

Hou, S.; Saas, A.; Chen, L.; and Ye, Y. 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)*, 104–111. IEEE.

Hou, S.; Ye, Y.; Song, Y.; and Abdulhayoglu, M. 2017. Hindroid: An intelligent android malware detection system based on structured heterogeneous information network. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1507–1515.

Hu, L.; Xu, S.; Li, C.; Yang, C.; Shi, C.; Duan, N.; Xie, X.; and Zhou, M. 2020. Graph Neural News Recommendation with Unsupervised Preference Disentanglement. In *ACL*, 4255–4264.

Kim, T.; Kang, B.; Rho, M.; Sezer, S.; and Im, E. G. 2018. A multimodal deep learning method for android malware detection using various features. *IEEE Transactions on Information Forensics and Security* 14(3): 773–788.

Liu, Y.; Wang, X.; Wu, S.; and Xiao, Z. 2019. Independence Promoted Graph Disentangled Networks. *arXiv preprint arXiv:1911.11430* .

Liu, Z.; Zhang, H.; Chen, Z.; Wang, Z.; and Ouyang, W. 2020. Disentangling and Unifying Graph Convolutions for Skeleton-Based Action Recognition. In *CVPR*, 143–152.

Locatello, F.; Bauer, S.; Lucic, M.; Raetsch, G.; Gelly, S.; Schölkopf, B.; and Bachem, O. 2019. Challenging Common Assumptions in the Unsupervised Learning of Disentangled Representations. In *ICML*, 4114–4124.

Ma, J.; Cui, P.; Kuang, K.; Wang, X.; and Zhu, W. 2019a. Disentangled graph convolutional networks. In *ICML*, 4212–4221.

Ma, J.; Zhou, C.; Cui, P.; Yang, H.; and Zhu, W. 2019b. Learning disentangled representations for recommendation. *arXiv preprint arXiv:1910.14238* .

Maaten, L. v. d.; and Hinton, G. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research* 9(Nov): 2579–2605.

Perozzi, B.; Al-Rfou, R.; and Skiena, S. 2014. Deepwalk: Online learning of social representations. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 701–710.

Sun, Y.; Han, J.; Yan, X.; Yu, P. S.; and Wu, T. 2011. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *VLDB Endowment* 4(11): 992–1003.

Tang, J.; Qu, M.; Wang, M.; Zhang, M.; Yan, J.; and Mei, Q. 2015. Line: Large-scale information network embedding. In *International Conference on World Wide Web*, 1067–1077.

Tran, L.; Yin, X.; and Liu, X. 2017. Disentangled representation learning gan for pose-invariant face recognition. In *CVPR*, 1415–1424.

Wang, X.; Jin, H.; Zhang, A.; He, X.; Xu, T.; and Chua, T.-S. 2020. Disentangled Graph Collaborative Filtering. In *SIGIR*, 1001–1010.

Ye, Y.; Hou, S.; Chen, L.; Lei, J.; Wan, W.; Wang, J.; Xiong, Q.; and Shao, F. 2019. Out-of-sample Node Representation Learning for Heterogeneous Graph in Real-time Android Malware Detection. In *IJCAI*, 4150–4156.

Ye, Y.; Li, T.; Adjeroh, D.; and Iyengar, S. S. 2017. A survey on malware detection using data mining techniques. *ACM Computing Surveys* 50(3): 1–40.

Ye, Y.; Wang, D.; Li, T.; Ye, D.; and Jiang, Q. 2008. An intelligent PE-malware detection system based on association mining. *Journal in computer virology* 4(4): 323–334.