Gotcha - Sly Malware! *Scorpion*: A Metagraph2vec Based Malware Detection System

Yujie Fan, Shifu Hou
Department of CSEE
West Virginia Univerisity, WV, USA
{yf0004,shhou}@mix.wvu.edu

Yiming Zhang, Yanfang Ye *
Department of CSEE
West Virginia Univerisity, WV, USA
yanfang.ye@mail.wvu.edu

Melih Abdulhayoglu
Comodo Security Solutions, Inc.
Clifton, NJ, USA
melih@comodo.com

ABSTRACT

Due to its severe damages and threats to the security of the Internet and computing devices, malware detection has caught the attention of both anti-malware industry and researchers for decades. To combat the evolving malware attacks, in this paper, we first study how to utilize both content- and relation-based features to characterize sly malware; to model different types of entities (i.e., *file*, *archive*, *machine*, *API*, *DLL*) and the rich semantic relationships among them (i.e., *file-archive*, *file-machine*, *file-file*, *API-DLL*, *file-API* relations), we then construct a structural heterogeneous information network (HIN) and present meta-graph based approach to depict the relatedness over files. To measure the relatedness over files on the constructed HIN, since malware detection is a cost-sensitive task, it calls for efficient methods to learn latent representations for HIN. To address this challenge, based on the built meta-graph schemes, we propose a new HIN embedding model ***metagraph2vec*** on the first attempt to learn the low-dimensional representations for the nodes in HIN, where both the HIN structures and semantics are maximally preserved for malware detection. A comprehensive experimental study on the real sample collections from Comodo Cloud Security Center is performed to compare various malware detection approaches. The promising experimental results demonstrate that our developed system ***Scorpion*** which integrate our proposed method outperforms other alternative malware detection techniques. The developed system has already been incorporated into the scanning tool of Comodo Antivirus product.

CCS CONCEPTS

• **Artifficial Intelligence** → **General**; • **Database applications** → *Data mining*; • **Security and Protection** → Invasive Software;

KEYWORDS

Malware Detection; Heterogeneous Information Network; Network Embedding; Metagraph2vec.

ACM Reference Format:
Yujie Fan, Shifu Hou, Yiming Zhang, Yanfang Ye *, and Melih Abdulhayoglu. 2018. Gotcha - Sly Malware! *Scorpion*: A Metagraph2vec Based Malware Detection System. In *KDD '18: The 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, August 19–23, 2018, London, United Kingdom.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3219819.3219862

KDD '18, August 19–23, 2018, London, United Kingdom

ACM ISBN 978-1-4503-5552-0/18/08…$15.00
https://doi.org/10.1145/3219819.3219862

1 INTRODUCTION

As the Internet and computing devices become increasingly ubiquitous, their security has become more and more important. Malware (short for ***mal***icious soft***ware***) is software designed to infiltrate or damage a computing system without the owner's informed consent [40], such as viruses, worms, trojans, bots, and ransomware. Malware has been used as a major weapon by cybercriminals to launch a wide range of security attacks, like stealing confidential information, hijacking computing devices remotely to deliver massive spam emails, and crippling critical infrastructures, which cause serious damages and significant financial loss to legitimate users [40]. For example, a recent study [29] showed that half a billion personal records were stolen or lost because of malware infections; it's also reported that the global damage costs of ransomware - a type of malware that threatens to publish the victim's data or perpetually block access to it unless a ransom is paid - exceeded $5 billion in 2017 [10]. As a result, the detection of malware is of major concern to both the anti-malware industry and researchers.

In order to protect legitimate users against malware attacks, the most significant line is anti-malware software products, such as Comodo, Kaspersky, and Symantec's Antivirus software. Traditionally, they mainly used the signature-based method [40] for detection. However, this method can be easily evaded by malware attackers using techniques such as encryption, obfuscation, and polymorphism [1]. To adjust this issue, anti-malware software products monitored behaviors of software from the Operating System (OS) kernel level to block malicious behaviors. Though dynamic detection is more resilient to low-level obfuscation, it is inherently cost expensive and does not scale well. Unfortunately, driven by the economic benefits, malicious programs are created and being disseminated at a rate of thousands per day [44], making it difficult for these client-based malware detection methods to be effective. In order to combat the malware attacks, intelligent malware detection systems have been developed at the cloud (server) side by applying data mining and machine learning techniques in recent years [12, 20, 36, 39, 41, 45, 46]. In many of these systems, based on the labeled files (either malicious or benign), different classification models were built resting on various kinds of content-based features, such as Windows Application Programming Interfaces (APIs), *n*-gram binaries, system calls that are either statically or dynamically extracted. The models in these systems merely utilizing file content have isolated successes in classifying particular sets of

malware samples, but ignoring the relations among file samples is a significant limitation of such kind of detection methods. Actually, the relations among file samples may provide invaluable information about their properties for malware detection. For example, if an unknown file (i.e., file without class label) always co-exists with many Trojans, then it's highly possible that this file is a malicious Trojan-downloader [44] - a type of malware that downloads and installs multiple unwanted software (e.g., Trojan, adware) from remote servers. To address the challenges of content-based malware detection, systems leveraging file relations (e.g., file-to-machine relations [3], file placements [32], file co-existences [30, 44]) have been developed for malware detection. However, these systems only took one single kind of file relations into consideration for malware detection.

Malware attack and defense are engaged in a never-ending arms race. The methods only utilizing either file content or file relations for detection open the possibility for malware attackers to come up with ways to bypass the detection. For example, as shown in Figure 1, to detect whether the newly collected file "*File-U: GWHookMan.dll*" is malicious or not, if we only count on the file content (e.g., Windows API calls which can well reflect the behaviors of program code), "*File-U*" is very similar to the benign file "*File-B1: AXSLE.dll*" which is a dynamic link library (DLL) file associated with Adobe Photoshop Elements 6; in this regard, it could be predicted as benign. If we merely depend on the file relations (e.g., file replacements which indicate files are compressed in the same archives like in the same .zips), "*File-U*" is always replaced in the same zip files together with a benign online game program "*File-B2: Lostsaga.exe*"; in this respect, it may also be predicted as benign. Actually, "*File-U*" is a malicious driver program associated with the online game Trojan "*File-M1: GameWatcher.exe*". Though from file content, "*File-U*" and "*File-M1*" are not directly related, the APIs they call have intrinsic connections. For example, the API of "SetTimer" called by "*File-M1*" and the API of "SetDoubleClickTime" called by "*File-U*" are both in the "USER32.DLL" and these two APIs are also always called together by other online game Trojans (e.g., "*File-M2: PK32.exe*"), as they are both related to the execution of online game acceleration. To catch sly malware like "*File-U*", it calls for novel representation to depict such complex relationships and also the detection model based on the representation.

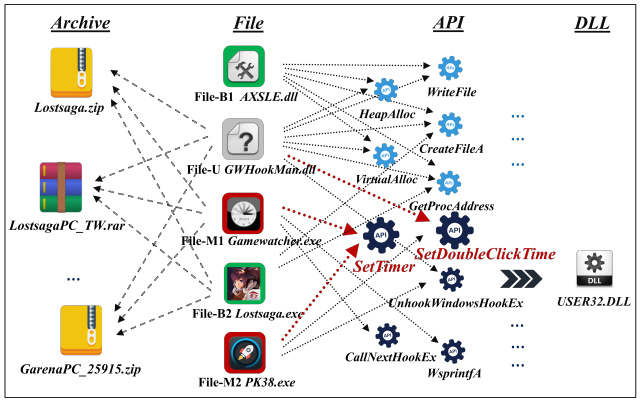


Figure 1: Catching sly malware from a comprehensive view.

To address the challenges above, in this paper, we leverage both content- and relation-based features to characterize the given Windows Portable Executable (PE) files, e.g., two files will be relevant if they are always replaced in the same archives and they have APIs that belong to the same DLL and are also called together by a set of other files. To capture such complex relationships, we first introduce a heterogeneous information network (HIN) [27] for representation and then use meta-graph based approach [48] to incorporate higher-level semantics to build up relatedness over the files. Since malware detection is a speed sensitive application and most network mining methods suffer the high computation and space cost, to tackle this problem, we leverage network embedding techniques and propose a new HIN embedding model named *metagraph2vec* to learn the low-dimensional representations for the nodes in HIN, where both the HIN structures and semantics are maximumly preserved. After that, a classifier will be built for malware detection. We develop a system called *Scorpion* integrating our proposed method for malware detection, which has the following major traits:

- ***Novel feature representation to enable the detection of sly malware:*** Instead of only using file content or file relations to describe the given files, we utilize both of them for malware detection. In *Scorpion*, the content-based features (e.g., Windows API calls) will be first extracted from the given files, and various kinds of relations will be further analyzed, such as (1) whether the files are replaced in the same archives, (2) whether the files co-exist in users' computing devices, (3) whether the files are created by the same parent files (e.g., the Instant Message (IM) software installation package "QQ9.0.1.exe" will create multiple files including "bugreport.exe" and "QQCloud.DLL" during the installation), (4) whether the extracted APIs belong to the same DLL (e.g., APIs of "WriteFile" and "CreateFileA" are both in "KERNEL32.DLL"), and (5) whether the APIs are called together by same set of files. Based on these extracted features, to model different kinds of entities (i.e., file, archive, machine, API, DLL) and the rich semantic relationships among them (i.e., *file-archive*, *file-machine*, *file-file*, *API-DLL*, *file-API* relations), a structural HIN is first introduced to represent the given files; and a meta-graph based approach is presented to characterize the relatedness over files. This is a natural way to handle different types of features, i.e., file content and file relations, but to our best knowledge is the first attempt to represent the files in such comprehensive way that is capable to provide a more resilient solution against malware's evasion tatcis.
- ***Efficient representation learning for HIN:*** Based on the built meta-graph schemes, a new HIN embedding model *metagraph2vec* is proposed on the first attempt to learn the low-dimensional representations for the nodes in HIN, which are capable to preserve both the semantics and structural correlations between different types of nodes. Though it's proposed for malware detection, the HIN embedding model *metagraph2vec* is a general framework which is able to learn desirable node representations in HIN and thus can be further applied to various network mining tasks, such as node classification, clustering and similarity search.
- ***A practical developed system for anti-malware industry application:*** We develop a practical system *Scorpion* for intelligent

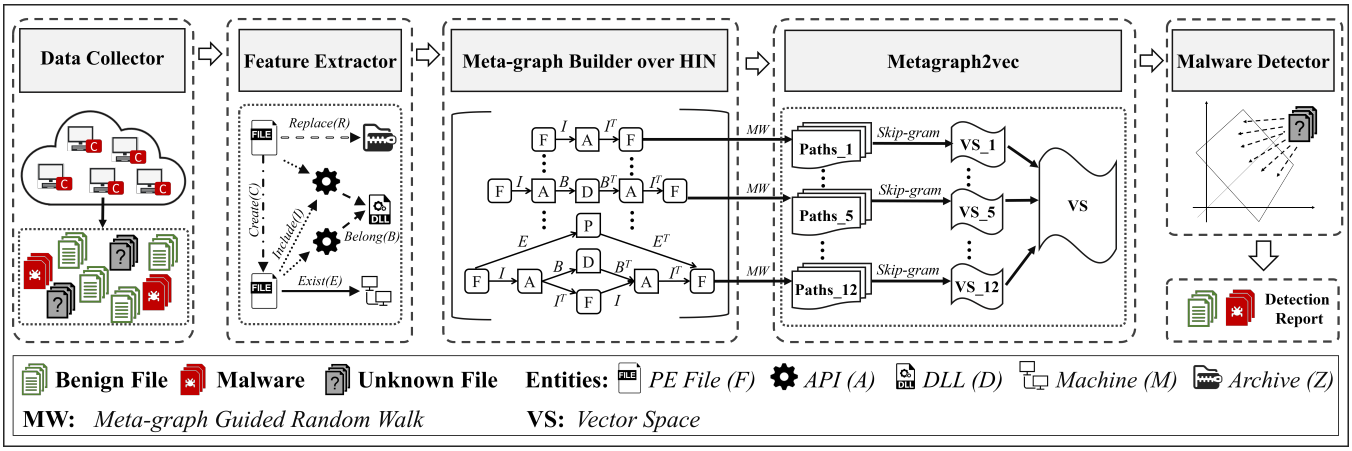


Figure 2: System architecture of *Scorpion*.

malware detection and provide a comprehensive experimental study based on the real sample collection from Comodo Cloud Security Center, which demonstrates the effectiveness and efficiency of our developed system. *Scorpion* has already been incorporated into the scanning tool of Comodo Antivirus product. It has been deployed and tested based on the real daily sample collection (over 500,000 PE files per day) for over a year.

The remainder of this paper is organized as follows. Section 2 introduces our system architecture. Section 3 presents our proposed method in detail. In Section 4, based on the real sample collection from Comodo Cloud Security Center, we systematically evaluate the performance of our developed system in comparisons with other alternative methods in malware detection. Section 5 presents the details of system development and operation. Section 6 discusses the related work. Finally, Section 7 concludes.

2 SYSTEM ARCHITECTURE

The architecture of our developed malware detection system *Scorpion* is shown in Figure 2, consisting of the following components:

- ***Data Collector.*** Through the installed Comodo Antivirus product, the users can upload the PE files to Comodo Cloud Security Center for detection. This module collects the uploaded files as well as the relations among files, machines, and archives.
- ***Feature Extractor.*** Based on the data collected from the previous module, it first extracts content-based features (e.g., Windows API calls) from the files, and then analyzes various relationships (i.e., *file-archive*, *file-machine*, *file-file*, *API-DLL*, *file-API* relations) among different types of entities (i.e., *file*, *archive*, *machine*, *API*, *DLL*) to depict the PE files. (See Section 3.1 for details.)
- ***Meta-graph Builder over HIN.*** In this module, based on the features extracted from the previous component, a structural HIN is first presented to model the relationships among different types of entities; and then different meta-graphs are built from the HIN to capture the relatedness over PE files from different views (i.e., with different semantic meanings). (See Section 3.2 for details.)
- ***Metagraph2vec.*** Based on the meta-graph schemes built from the previous module, a HIN embedding model *metagraph2vec* is proposed to learn the low-dimensional representations for the nodes in HIN, which are capable to preserve both the semantics and structural correlations between different types of nodes. In *metagraph2vec*, given a meta-graph scheme, a meta-graph guided random walk method is first proposed to map the word-context concept in a text corpus into a HIN, and then skip-gram is utilized to learn effective node representation for a HIN. Later, multi-view fusion algorithm is proposed to incorporate different representations learned based on different meta-graph schemes. (See Section 3.3 for details.)
- ***Malware Detector.*** After the latent representation learning for HIN using *metagraph2vec*, the mapped low-dimensional vectors of PE files will be fed to a Support Vector Machine (SVM) to train the classification model, based on which the unlabeled files can be predicted as either benign or malicious.

3 PROPOSED METHOD

In this section, we introduce the detailed approaches of how we represent the PE files utilizing both content- and relation-based features simultaneously, and how we solve the malware detection problem based on the representation.

3.1 Feature Extraction

Content-based Features. Since Windows API calls can effectively reflect the behaviors of PE program codes [45] (e.g., the API of "GetFileType" in "KERNEL32.DLL" can be used to retrieve the file type of the specified file, while the API of "GetDlgItemText" in "USER32.DLL" can be utilized to obtain the title or text associated with a control in a dialog box), we extract Windows API calls from the Import Tables [45, 46] of the collected PE files as content-based features to represent them. If a PE file is previously compressed by a third party binary compress tool such as UPX and ASPack Shell or embedded a homemade packer, it will be decompressed at first using CMDsm developed by Comodo Anti-malware Lab before feature extraction. Note that the content-based features of API calls are exploited as a case study here to facilitate the understanding of

our further proposed approach, while other features (e.g., n-gram binaries, dynamic system calls) either statically or dynamically extracted are also applicable in our further investigation.

Relation-based Features. Although content-based features like API calls can be used to represent the behaviors of a file, to catch sly malware like "*File-U: GWHookMan.dll*" as aforementioned, the intrinsic and complex relationships between it and its associated online game Trojan "*File-M1: GameWatcher.exe*" provide critical information for the detection. To detect the increasingly sophisticated malware, besides content-based features, the following kinds of relationships are also considered in our application.

- ***R1***: To describe the relation between a file and the archive it replaced in, we build the ***file-replace-archive*** matrix **R** where each element $r_{i,j} \in \{0, 1\}$ means if file i is replaced in archive j.
- ***R2***: To represent the file-machine relationship, we generate the ***file-exist-machine*** matrix **E** where each element $e_{i,j} \in \{0, 1\}$ denotes whether file i exists in machine j.
- ***R3***: A file can create another file during its execution. To represent such relationship between two files, we build the ***file-create-file*** matrix **C** where element $c_{i,j} \in \{0, 1\}$ denotes whether file i creates file j.
- ***R4***: To describe the relation of a file and its extracted API calls, we generate the ***file-include-API*** matrix **I** where each element $i_{i,j} \in \{0, 1\}$ denotes if file i includes API j.
- ***R5***: To denote the relation that an API belongs to a DLL, we generate the ***API-belongto-DLL*** matrix **B** where each element $b_{i,j} \in \{0, 1\}$ means if API i belongs to DLL j.

3.2 Meta-graph Based Relatedness

In order to depict PE files, archives, machines, APIs, DLLs and the rich relationships among them (i.e., ***R1***-***R5***), it is important to model them in a proper way so that different kinds of relations can be better and easier handled. We introduce how to use HIN, which is capable to be composed of different types of entities and relations, to represent the PE files by using the features described above. We first present the concepts related to HIN as follows.

Definition 3.1. ***Heterogeneous information network (HIN)*** [27]. A HIN is defined as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with an entity type mapping $\phi: \mathcal{V} \rightarrow \mathcal{A}$ and a relation type mapping $\psi: \mathcal{E} \rightarrow \mathcal{R}$, 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*** [28] for a HIN $\mathcal{G}$, 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 $\mathcal{R}$.

HIN not only provides the network structure of the data associations, but also provides a high-level abstraction of the categorical association. For our case, we have five entity types (i.e., PE file, archive, machine, API, DLL) and five types of relations among them (i.e., ***R1***-***R5***). Based on the definitions above, the network schema for HIN in our application is shown in Figure 3, which enables the representation of PE files in a comprehensive way that utilize both content- and relation-based information simultaneously.

The different types of entities and relations motivate us to use a machine-readable representation to enrich the semantics of relatedness among PE files. To handle this, the concept of meta-path has been proposed [28]: a meta-path $\mathcal{P}$ is a path defined on the graph of network schema $\mathcal{T}_{\mathcal{G}} = (\mathcal{A}, \mathcal{R})$, and is denoted in the form of $A_1 \xrightarrow{R_1} A_2 \xrightarrow{R_2} \ldots \xrightarrow{R_L} A_{L+1}$, which defines a composite relation $R = R_1 \cdot R_2 \cdot \ldots \cdot R_L$ between types A_1 and A_{L+1}, where $\cdot$ denotes relation composition operator, and L is the length of $\mathcal{P}$. A typical meta-path to formulate the relatedness over PE files by leveraging content-based features is $F \xrightarrow{includes} A \xrightarrow{includes^{-1}} F$ which means that two PE files can be connected through the path containing the same API over the HIN; while the meta-path of $F \xrightarrow{exists} M \xrightarrow{exists^{-1}} F$ denotes that two PE files can be connected through the path existing in the same machine over the HIN which is one formulation based on relation-based features. Although meta-path can be used to depict the relatedness over PE files in our application, it fails to capture a more complex relationship, such as the relation between "*File-U: GWHookMan.dll*" and its associated online game Trojan "*File-M1: GameWatcher.exe*" (i.e., they are related since they are always replaced in the same archives and they have APIs that are in the same DLL and are also called together by same set of other online game Trojans). This calls for a better characterization to handle such complex relationship. Meta-graph [48] is proposed to use a directed acyclic graph of entity and relation types to capture more complex relationship between two HIN entities, defined as follows:

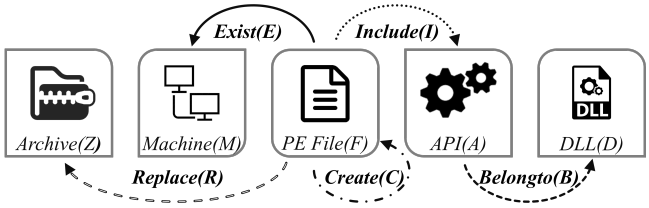


Figure 3: Network schema for HIN.

Definition 3.2. A ***meta-graph*** [48] $\mathcal{M}$ is a directed acyclic graph with a single source node n_s and a single target node n_t, defined on a HIN $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with schema $\mathcal{T}_{\mathcal{G}} = (\mathcal{A}, \mathcal{R})$. Formally, a meta-graph is defined as $\mathcal{M} = (\mathcal{V}_M, \mathcal{E}_M, \mathcal{A}_M, \mathcal{R}_M, n_s, n_t)$, where $\mathcal{V}_M \subseteq \mathcal{V}$, $\mathcal{E}_M \subseteq \mathcal{E}$ constrained by $\mathcal{A}_M \subseteq \mathcal{A}$ and $\mathcal{R}_M \subseteq \mathcal{R}$, respectively.

In our application, based on the HIN schema displayed in Figure 3, incorporated the domain knowledge from human experts in Comodo Anti-malware Lab, we generate twelve meaningful meta-graphs to characterize the relatedness over PE files (i.e., ***MID1–MID12*** shown in Figure 4). In fact, a meta-path is a special case of a meta-graph (e.g., *MID2* and *MID3* are particular cases of *MID7*). But meta-graph is capable to express more complex relationship in a convenient way. Different meta-graphs measure the relatedness between two PE files at different views. For example, ***MID7*** depicts that two PE files are related if they co-exist at the same machines and are also replaced in the same archives; while ***MID10*** describes that two PE files are connected if they are replaced in the same archives and they have APIs that are in the same DLL and are also called together by same set of other PE files (e.g., the relations between "*File-U: GWHookMan.dll*" and its associated online game Trojan "*File-M1: GameWatcher.exe*" can be well described using this

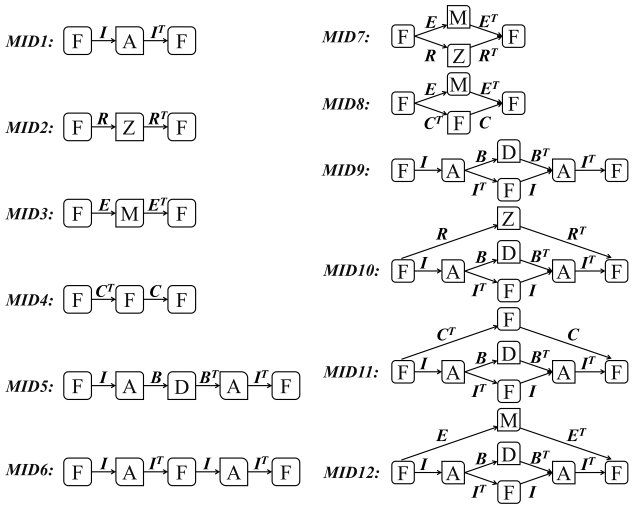


Figure 4: Meta-graphs built for malware detection. (The symbols in the figure are abbreviations shown in Figure 3.)

meta-graph). To measure the relatedness over HIN entities (e.g., PE files), traditional representation learning for HIN [17, 28, 37, 48] mainly focuses on factorizing the matrix (e.g., adjacency matrix) of a HIN to generate latent-dimension features for the nodes in this HIN. However, the computational cost of decomposing a large-scale matrix is usually very expensive, and also suffers from its statistical performance drawback [15]. Since malware detection is a speed sensitive application and requires cost-effective solutions, scalable representation learning method for HIN is in need.

3.3 Metagraph2vec

To address the above challenge, we first formalize the problem of HIN representation learning as follow.

Definition 3.3. ***HIN Representation Learning*** [11, 14]. Given a HIN $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, the representation learning task is to learn a function $f : \mathcal{V} \rightarrow \mathbb{R}^d$ 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 capable to preserve the structural and semantic relations among them.

To solve the problem of HIN representation learning, due to the heterogeneous property of HIN (i.e., network consisting of multi-typed entities and relations), it is difficult to directly apply the conventional homogeneous network embedding techniques (e.g., DeepWalk [24], LINE [31], node2vec [15]) to learn the latent representations for HIN. To address this issue, metapath2vec [11] was proposed which employed meta-path based random walks and heterogeneous skip-grams to learn the latent representations for HIN such that the semantic and structural correlations between different types of nodes could be persevered. In our application, to catch sly malware like "*File-U: GWHookMan.dll*", meta-graph based scheme is designed to capture the relatedness over PE files which is more expressive than meta-path based approach to depict the complex relationships among PE files. This calls for a new method to learn the latent representations for HIN. To tackle this problem, we propose a new HIN embedding framework *metagraph2vec* to learn desirable node representations in HIN: first, meta-graph guided random walk is proposed to map the word-context concept in a text corpus into a HIN; then skip-gram is utilized to learn effective node representation for a HIN; later, a multi-view fusion algorithm is proposed to incorporate different node representations learned based on different meta-graph schemes.

Meta-graph Guided Random Walk. Given a source node v_j in a homogeneous network, the traditional random walk is a stochastic process with random variables $v_j^1, v_j^2, ..., v_j^k$ such that v_j^{k+1} is a node chosen at random from the neighbors of node v_k. The transition probability $p(v_j^{i+1}|v_j^i)$ at step i is the normalized probability distributed over the neighbors of v_j^i by ignoring their node types. However, this mechanism is unable to capture the semantic and structural correlations among different types of nodes in a HIN. Here, we show how we use meta-graph to guide random walkers in a HIN to generate the paths of multiple types of nodes. In our application, given a HIN $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with schema $\mathcal{T}_\mathcal{G} = (\mathcal{A}, \mathcal{R})$, and a meta-graph scheme $\mathcal{M}$ in the basic form: $A_1 \begin{smallmatrix}\nearrow A_2 \searrow \\ \searrow A_3 \nearrow\end{smallmatrix} A_4$, we put two random walkers, i.e., *walker u* and *walker v*, to traverse the HIN. The transition probabilities at step i are defined as follows:

$$p_1(u^{i+1}|u^i, \mathcal{M}), p_2(v^{i+1}|v^i, \mathcal{M})$$

$$= \begin{cases} \frac{1}{|N_{A_t}(u^i)|}, \frac{1}{|N_{A_{t'}}(v^i)|} \\ \qquad \text{if}(u^i, u^{i+1}) \in E, (v^i, v^{i+1}) \in E, \phi(u^{i+1}) \neq \phi(v^{i+1}) \\ \frac{1}{|N_{A_t}(u^i) \cap N_{A_{t'}}(v^i)|}, \frac{1}{|N_{A_t}(u^i) \cap N_{A_{t'}}(v^i)|} \\ \qquad \text{if}(u^i, u^{i+1}) \in E, (v^i, v^{i+1}) \in E, u^{i+1} = v^{i+1} \\ 0, 0 \qquad \text{otherwise,} \end{cases} \tag{1}$$

where ϕ is the node type mapping function, $N_{A_t}(u^i)$ and $N_{A_{t'}}(v^i)$ denote A_t and $A_{t'}$ type of neighborhood of node u^i and v^i respectively. Note that if *walker u* and *walker v* are the same one, the meta-graph guided random walk can be considered as meta-path based random walk. The paths generated by the proposed meta-graph guided random walks are able to preserve both the semantic and structural relations between different types of nodes, and thus will facilitate the transformation of HIN structures into skip-gram.

Skip-gram. After mapping the word-context concept in a text corpus into a HIN via meta-graph guided random walk strategy (i.e., a sentence in the corpus corresponds to a sampled path and a word corresponds to a node), skip-gram [21] is then applied on the paths to maximize the probability of observing a node's neighbourhood (within a window w) conditioned on its current representation. The objective function of skip-gram is:

$$\arg\max_Y \log \sum_{-w \leq k \leq w, j \neq k} p(v_{j+k}|Y(v_j)), \tag{2}$$

where $Y(v_j)$ is the current representation vector of v_j, $p(v_{j+k}|Y(v_j))$ is defined using the softmax function:

$$p(v_{j+k}|Y(v_j)) = \frac{\exp(Y(v_{j+k}) \cdot Y(v_j))}{\sum_{q=1}^{|V|} \exp(Y(v_q) \cdot Y(v_j))}. \tag{3}$$

Due to its efficiency, we first apply hierarchical softmax technique [22] to solve Eq. 3; then the stochastic gradient descent (SGD) [2] is used to train the skip-gram.

Multi-view Fusion. Given a meta-graph scheme, by using the above proposed meta-graph guided random walk strategy and skip-gram, the node representations will be learned for a HIN. In our application, as described in Section 3.2, we have twelve meaningful meta-graphs (i.e., ***MID1–MID12***). Different meta-graphs characterize the relatednesses over PE files at different views, i.e., with different semantic meanings. For instance, ***MID1*** depicts the relatedness over PE files through their content correlations (i.e., the APIs they call); ***MID3*** describes the relatedness over PE files through the file-machine relations (i.e., the machines they co-exist); while ***MID10*** poetries the relatedness over PE files from a more comprehensive view which incorporates their content correlations and different intrinsic connections (i.e., they are connected as they are replaced in the same archives, and some APIs they call are from same DLLs and also called by other files). As different mata-graphs depict the relatedness over PE files in very diverse ways, to explore the complementary nature of these different views, we propose to use a multi-view fusion algorithm to incorporate different node representations learned based on different meta-graph schemes.

Given m kinds of node representations $Y_i (i = 1, ..., m)$ learned based on m meta-graph schemes (in our case $m = 12$), the incorporated node representations can be denoted as: $Y' = \alpha_i \times Y_i$, where $\alpha_i (i = 1, ..., m)$ is the weight of Y_i. To determine the weight of α_i for each mapped low-dimensional vector space Y_i, we measure the geometric distances among them. The distance measure based on the principal angles between two vector spaces is significant if and only if the vector spaces have the same dimensions [53]. In our case, the m mapped vector spaces are all with the same dimensions of d. Therefore, we apply the geodesic distance based on principal angles [19] to measure the geometric distances between the mapped vector spaces. The principal angle between space Y_i and Y_j is defined as the number $0 \leq \theta \leq \frac{\pi}{2}$ that satisfies:

$$\cos\theta = \max_{\mathbf{y}\in Y_i, \mathbf{y'}\in Y_j} \mathbf{y}^T \mathbf{y'}. \tag{4}$$

The angle θ is 0 if and only if $Y_i \bigcap Y_j \neq 0$, while $\theta = \frac{\pi}{2}$ if and only if $Y_i \perp Y_j$. Let $\theta_1, \theta_2, ..., \theta_d$ be the d principal angles between space Y_i and Y_j, the geodesic distance between them is formulated as:

$$d(Y_i, Y_j) = \sqrt{\theta_1^2 + \theta_2^2 + ... + \theta_d^2}. \tag{5}$$

Thus, we compute α_i for each mapped vector space Y_i as:

$$\alpha_i = \frac{\sum_{j=1, i\neq j}^{m} d(Y_i, Y_j)}{\sum_{i=1}^{m}\sum_{j=1, i\neq j}^{m} d(Y_i, Y_j)} \tag{6}$$

4 EXPERIMENTAL RESULTS AND ANALYSIS

In this section, we conduct five sets of experimental studies using real sample collections from Comodo Cloud Security Center to fully evaluate the performance of our developed malware detection system *Scorpion* with integrate the above proposed method.

4.1 Experimental Setup

Through the installed Comodo Antivirus product, its users can scan the PE files in their computing devices and upload those "suspicious" files (i.e., may always include sophisticated and evolving malware) recognized by the Comodo Antivirus in the client side to the cloud for further detection. We obtain the dataset from Comodo Cloud Security Center, which contains 59,749 PE files uploaded by 3,782 users (i.e. machines) on Jan 15, 2018 (i.e., 14,285 of them are detected as malicious in the cloud side, 21,429 of them are detected as benign, and 24,035 of them are unknown that require further prediction). We use those ones detected in the cloud as training data (i.e., 35,714 PE files: 14,285 malware and 21,429 benign files) and the 24,035 unknown files as testing data (to obtain the ground truth, they are further analyzed by the anti-malware experts of Comodo Security Lab, 8,893 of which are labeled as malicious and 15,142 are benign). After feature extraction and based on the designed network schema, the constructed HIN has 613,581 nodes (i.e., 59,749 nodes with type of PE file, 3,782 nodes with type of machine, 75,274 nodes with type of archive, 468,074 with types of API, 6,702 nodes with type of DLL) and 11,479,527 edges including relations of ***R1***-***R5***. To quantitatively validate the malware detection effectiveness, we use the performance measures shown in Table 1.

Table 1: Performance indices of malware detection.

Indices	Description
TP	# of PE files correctly classified as malicious
TN	# of PE files correctly classified as benign
FP	# of PE files mistakenly classified as malicious
FN	# of PE files mistakenly classified as benign
Precision	$TP/(TP + FP)$
Recall	$TP/(TP + FN)$
ACC	$(TP + TN)/(TP + TN + FP + FN)$
F1	$2 * Precision * Recall/(Precision + Recall)$

Table 2: Detection Results of different meta-graphs.

ID	Meta-paths included	Precision	Recall	ACC	F1
MID1	-	0.740	0.820	0.827	0.778
MID2	-	0.679	0.770	0.780	0.722
MID3	-	0.608	0.713	0.724	0.657
MID4	-	0.664	0.759	0.768	0.708
MID5	-	0.755	0.831	0.837	0.791
MID6	-	0.757	0.831	0.839	0.792
MID7	MID2&MID3	0.698	0.788	0.795	0.740
MID8	MID3&MID4	0.704	0.783	0.798	0.741
MID9	MID5&MID6	0.771	0.845	0.849	0.806
MID10	MID2&MID5&MID6	0.808	0.865	0.874	0.835
MID11	MID4&MID5&MID6	0.791	0.852	0.862	0.820
MID12	MID3&MID5&MID6	0.786	0.854	0.860	0.819

4.2 Evaluation of Different Meta-graphs

In this set of experiments, based on the dataset described in Section 4.1, we evaluate the performance of different kinds of relatedness over PE files depicted by different meta-graphs (i.e., ***MID1-MID12***). In the experiments, given a meta-graph scheme, we use the meta-graph guided random walk method and skip-gram described in

Table 3: Comparisons of *metagraph2vec* with other network representation learning methods in malware detection.

Metric	Method	Training									Testing
		10%	20%	30%	40%	50%	60%	70%	80%	90%	
ACC	DeepWalk	0.742	0.800	0.841	0.858	0.854	0.867	0.887	0.898	0.901	0.907
	LINE	0.778	0.858	0.894	0.917	0.930	0.933	0.942	0.946	0.944	0.948
	metapath2vec	0.819	0.880	0.914	0.932	0.935	0.949	0.953	0.954	0.959	0.957
	metagraph2vec	0.854	0.906	0.921	0.945	0.939	0.958	0.970	0.977	0.978	0.983
F1	DeepWalk	0.741	0.802	0.840	0.857	0.854	0.867	0.887	0.897	0.907	0.877
	LINE	0.781	0.867	0.900	0.921	0.932	0.935	0.944	0.947	0.945	0.930
	metapath2vec	0.817	0.881	0.913	0.932	0.934	0.948	0.954	0.953	0.958	0.942
	metagraph2vec	0.853	0.907	0.920	0.945	0.939	0.957	0.970	0.976	0.977	0.977

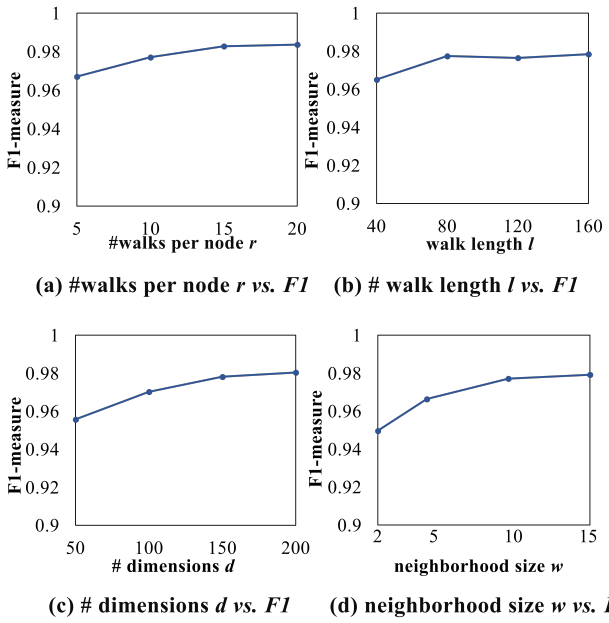
Section 3.3 to learn the low-dimensional representations of the nodes with type of PE file in the HIN, which are then fed to SVM to build the classification model for malware detection. For SVM, we use LibSVM and the penalty is empirically set to be 1,000 while other parameters are set by default. The experimental results are shown in Table 2, from which we can see that different meta-graphs show different performances in malware detection, since each of them represents a specific semantic in malware detection domain. From Table 2, we can also observe that: (1) the relatedness over PE files depicted by content-based correlations (***MID1***: APIs called by the files) performs better than single kind of relation-based correlations (relations of ***MID2***: *file-archive*, ***MID3***: *file-machine*, ***MID4***: *file-file*); (2) meta-graph based scheme is more expressive than meta-path based approach in depicting more complex and comprehensive relationships among PE files and thus achieve better detection performance: i) for the description of content-based correlations, ***MID9*** outperforms *MID1*, *MID5* and *MID6*, ii) for the description of relation-based correlations, ***MID7*** and ***MID8*** outperforms *MID2*, *MID3* and *MID4*; iii) obviously, the relationships among PE files depicted by the meta-graphs consisting of both content- and relation-based correlations (i.e., ***MID10-MID12***) provide much higher-level semantics than others, based on which the detection is significantly better than the others (i.e., MID1-MID9). It will be interested to see the performance if different meta-graphs are incorporated together for the detection. This will be evaluated in the next set of experiments.

4.3 Evaluation of Metagraph2vec

In this set of experiments, we evaluate our proposed method *metagraph2vec* by comparisons with several recent network representation learning methods: DeepWalk [24], LINE [31] and metapath2vec [11]. For DeepWalk and LINE, we ignore the heterogeneous property of HIN and directly feed the HIN for representation learning; for metapath2vec, since *MID1-MID6* are special meta-graphs which can also be considered as meta-paths, we use them to guide the random walks in metapath2vec. The parameter settings used for *metagraph2vec* are in line with typical values used for DeepWalk, LINE and metapath2vec: vector dimension $d = 128$ (LINE: 128 for each order (1st- and 2nd-order)), walks per node $r = 10$, walk length $l = 80$ and window size $w = 10$. To facilitate the comparisons, we use the experimental procedure as in [11, 24, 31]: we randomly select a portion of training data described in Section 4.1 (ranging from 10% to 90%) for training and the remaining ones for testing. We also evaluate their performance on our testing dataset described in Section 4.1 using all the training data to train. The SVM is used as the classification model for all the methods. Table 3 illustrates the detection results of different network representation learning methods. From Table 3, we can see that the proposed *metagraph2vec* model consistently and significantly outperforms all baselines for malware detection in terms of *ACC* and *F1*. That is to say, *metagraph2vec* learns significantly better file representations than current state-of-the-art methods. The success of *metagraph2vec* lies in the proper consideration and accommodation of the heterogeneous property of HIN (i.e., the multiple types of nodes and relations), and the advantage of meta-graph guided random walk for sampling the node paths. Furthermore, from Table 2 and Table 3, we can also observe that, compared with any node representations learned based on individual meta-graph scheme (i.e., *MID1-MID12*), using the multi-view fusion algorithm proposed in Section 3.3 to incorporate different node representations learned based on different meta-graph schemes can significantly improve the detection performance.

4.4 Evaluation of Parameter Sensitivity, Scalability and Stability

In this set of experiments, based on the dataset described in Section 4.1, we first conduct the **sensitivity** analysis of how different choices of parameters will affect the performance of *metagraph2vec* in malware detection. From the results shown in Figure 5(a) and 5(b), we can observe that the balance between computational cost (number of walks per node r and walk length l in x-axis) and efficacy (F1 in y-axis) can be achieved when $r = 15$ and $l = 80$ for malware detection. We also examine how latent dimensions (d) and neighborhood size (w) affect the performance. As shown in Figure 5(c), we can see that the performance tends to be stable once d reaches around 150; similarly, from Figure 5(d) we can find that the performance inclines to be stable when w increases to 10. Overall, *metagraph2vec* is not strictly sensitive to these parameters and is able to reach high performance under a cost-effective parameter choice. We then further evaluate the **scalability** of *metagraph2vec* which can be parallelized for optimization. We run the experiments using the default parameters with different number of threads (i.e., 1, 4, 8, 12, 16), each of which utilizes one CPU core. Figure 6.*left*



(a) #walks per node *r* *vs. F1* (b) # walk length *l* *vs. F1*

(c) # dimensions *d* *vs. F1* (d) neighborhood size *w* *vs. F1*

Figure 5: Parameter sensitivity evaluation.

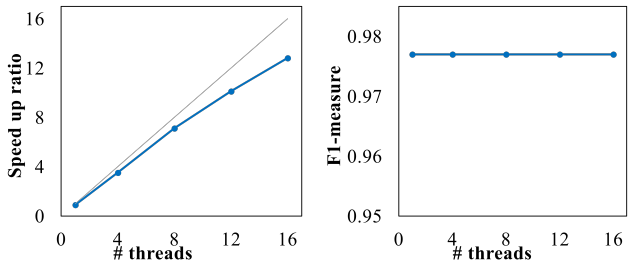


Figure 6: *Left*: Speed-up vs # threads, *Right*: F1 vs # threads.

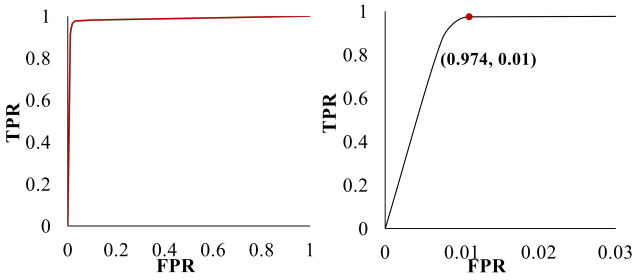


Figure 7: ROC curves: *Left*: Training, *Right*: Testing.

shows the speed-up of *metagraph2vec* deploys multiple threads over the single-threaded case, which shows that the model achieves acceptable sub-linear speed-ups as the line is close to the optimal line; while Figure 6.*right* shows that the performance remains stable when using multiple threads for model updating. Overall, the proposed system are efficient and scalable for large-scale HIN with large numbers of nodes. For **stability** evaluation, Figure 7 shows the training (i.e., ten-fold cross validations) and testing receiver operating characteristic (ROC) curves; it achieves an impressive 0.974 TP rate at the 0.01 FP rate when labeling the unknown files in the testing dataset.

4.5 Comparisons with Other Traditional Machine Learning Methods

In this set of experiments, based on the dataset described in Section 4.1, we compare *Scorpion* which integrates our proposed method described in Section 3 with other traditional machine learning methods. For these methods, we construct three types of features: ***f-1***: content-based features (i.e., API calls); ***f-2***: three original relation-based features (i.e., ***R1-R3*** introduced in Section 3.1); ***f-3***: augmented features of API calls and ***R1-R3***. Based on these features, we consider two typical classification models, i.e., Naive Bayes (NB) and SVM. The experimental results are illustrated in Table 4. From the results we can observe that feature engineering (***f-3***: concatenation of different features altogether) helps the performance of machine learning, but *Scorpion* added the knowledge represented as HIN significantly outperforms other baselines. This again demonstrates that, to detect the increasingly sophisticated malware, *Scorpion* using meta-graph based approach over HIN is able to build the higher-level semantic and structural connection between PE files with a more expressive and comprehensive view and thus achieves better detection performance.

Table 4: Comparisons of other machine learning methods.

Method	NB			SVM			*Scorpion*
Settings	***f-1***	***f-2***	***f-3***	***f-1***	***f-2***	***f-3***	/
ACC	0.795	0.778	0.830	0.826	0.812	0.873	**0.983**
F1	0.742	0.720	0.783	0.778	0.762	0.836	**0.977**

4.6 Comparisons with Anti-malware Products

In this section, we evaluate the detection performance of our developed system *Scorpion* in comparisons with some popular Anti-malware products, based on the labeled 8,893 malware in the testing dataset. To conduct the experiments, we use Kaspersky (K): 18.0.0.405, McAfee (M): 10.5.3, Symantec (S): 22.11.0.41, TrendMicro (T): 11.0, and VirusTotal[1] for comparisons. VirusTotal (VT) is a free service that analyzes uploaded files and URLs and facilitates the quick detection of malware, which aggregates more than 60 anti-malware scanning engines (if there are $\geq 1/3$ of the anti-malware vendors in VT detecting the file as malicious, then it will be considered as malware for comparisons). The comparison results are shown in Table 5, from which we can see that *Scorpion* outperforms others (with TPR of 97.4%) in detecting newly unleashed malware from different families (e.g., Locky, Ramnit, Online Game Trojans, etc.). The success of *Scorpion* may lie in its novel higher-level semantic and comprehensive portrait of PE files as well as the effective HIN representation learning using the proposed *metagraph2vec*.

[1]https://www.virustotal.com/

Table 5: Comparisons with different anti-malware scanners.

Family	File#	K	M	S	T	VT	**Scorpion**
Locky	676	656	648	650	652	651	674
Pushdo	651	643	640	642	644	642	649
Ramnit	544	529	528	531	528	526	542
Zeus	418	408	392	403	396	391	413
Fireball	374	369	363	366	367	365	373
Conficker	363	358	356	359	354	357	361
PSWTroj	249	242	237	235	239	234	246
Tinba	181	174	170	173	170	171	179
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Others	2332	2302	2295	2300	2297	2295	2318
Total	8893	8235	8101	8217	8163	8457	8666
DetectionRate	-	0.926	0.911	0.924	0.918	0.951	**0.974**

5 SYSTEM DEPLOYMENT AND OPERATION

By the date, our developed system *Scorpion* which integrates our proposed method, has already been incorporated into the scanning tool of Comodo's Antivirus product. *Scorpion* has been used to predict the daily sample collection from Comodo Cloud Security Center which contains over 500,000 newly collected PE files per day. Note that malware are constantly evolving and new malware samples are produced on a daily basis. To account for the temporal trends of malware production, the training data of our developed system *Scorpion* are dynamically changing to include newly collected samples and information. Our system *Scorpion* has been deployed and tested based on the real daily sample collection for over a year.

For the development of the system, Comodo has spent over $300K, $150K of which is on the hardware equipment. Due to the high detection efficiency and effectiveness, *Scorpion* can greatly save human labors and reduce the staff cost: over 60 anti-malware analysts at Comodo Cloud Security Center are utilizing the system on a daily basis. In practice, an anti-malware analyst has to spend at least 8 hours to manually analyze 100 PE files for malware detection. Using the developed system *Scorpion*, the analysis of about 500,000 PE files (including feature extraction and prediction) can be performed within hours using multiple servers. This would benefit over 12 million Comodo's Antivirus product users.

6 RELATED WORK

In recent years, there have been many research efforts on developing intelligent malware detection systems using machine learning and data mining techniques [3, 5, 6, 20, 30, 42, 43, 45, 46]. Most of them utilize either content-based features or relation-based features for detection. For content-based detection [7, 12, 20, 36, 38, 39, 41, 45], various kinds of content features such as API calls, *n*-gram binaries, system calls either statically or dynamically extracted are used to capture the characteristics of PE files; while relation-based detection methods [3, 4, 6, 23, 30] leverage the relations such as file-file relations, file-machine relations for feature representation. Actually, combining both content- and relation-based features can achieve better performance in malware detection. In our previous work [44], we proposed a semi-parametric classification model to combine both file content and file co-existences for malware detection; in our another recent work [18], HIN was first time introduced to model the files, APIs and the relationships among them, and then meta-paths over HIN were built to formulate the relatedness over files for malware detection. Unfortunately, malware attack and defense are engaged in a never-ending arms race. As meta-path structure fails to depict the more complex relationships like relations between "*File-U: GWHookMan.dll*"and its associated online game Trojan "*File-M1: GameWatcher.exe*", to catch sly malware like "*File-U: GWHookMan.dll*", in this paper, we propose to use meta-graph over HIN to capture a more comprehensive relatedness over PE files for malware detection.

HIN has been intensively studied in recent years. Typically, HIN is used to model different types of entities and relations [16, 25]. It has been applied to various applications, such as scientific publication network analysis [26, 28], document analysis based on knowledge graph [33, 34], social network analysis for Twitter users [13, 47, 49–52], and malware detection [18]. Several measures (e.g., meta-path [28] and meta-graph [48]) have already been proposed for relevance computation over HIN entities. However, as malware detection requires cost-effective solutions, it calls for efficient methods for HIN representation learning. Recently, many efficient network embedding methods [8, 9, 35] have been proposed to address representation learning for homogeneous network, such as DeepWalk [24], node2vec [15], and LINE [31]. However, due to the heterogeneous properties of HIN, it's difficult to directly apply them for HIN representation learning. To tackle this challenge, metapath2vec [11], HIN2vec [14] have been proposed for HIN representation learning, which are all based on meta-path scheme. In our application, a new efficient HIN representation learning method based on meta-graph scheme is in demand. To address this issue, we propose *metagraph2vec* to learn the latent representations for HIN which is capable to preserve both the semantics and structural correlations among different types of nodes in HIN.

7 CONCLUSION

To combat the evolving malware attacks, in this paper, we first study how to utilize both content- and relation-based features to characterize sly malware and then develop an intelligent system *Scorpion* for its detection. In *Scorpion*, to model different types of entities (i.e., file, archive, machine, API, DLL) and the rich semantic relationships among them (i.e., *file-archive*, *file-machine*, *file-file*, *API-DLL*, *file-API* relations), a structural HIN is first introduced to represent the given files; and then meta-graph based approach is presented to depict the relatedness over files. To reduce the high computation cost of representation learning for HIN, in *Scorpion*, a new HIN embedding model *metagraph2vec* is proposed on the first attempt to learn the low-dimensional representations for the nodes in HIN based on meta-graph schemes. A comprehensive experimental study on the real sample collections from Comodo Cloud Security Center is performed to compare various malware detection approaches. The promising experimental results demonstrate that *Scorpion* outperforms other alternative malware detection methods as well as popular Antivirus products. The system has already been incorporated into the scanning tool of Comodo Antivirus product.

ACKNOWLEDGMENTS

The authors would like to thank the anti-malware experts of Comodo Security Lab for the data collection as well as helpful discussions and supports. This work is also supported by the U.S. National Science Foundation (CNS-1618629), WV HEPC Grant (HEPC.dsr.18.5) and WVU Research and Scholarship Advancement Grant (844).

REFERENCES

[1] Philippe Beaucamps and Éric Filiol. 2007. On the possibility of practically obfuscating programs towards a unified perspective of code protection. *Journal in Computer Virology* 3, 1 (2007), 3–21.
[2] Léon Bottou. 1991. Stochastic gradient learning in neural networks. *Proceedings of Neuro-Nımes* 91, EC2 (1991).
[3] Duen Horng Chau, Carey Nachenberg, Jeffrey Wilhelm, Adam Wright, and Christos Faloutsos. 2010. Polonium: Tera-scale graph mining for malware detection. In *KDD*.
[4] Lingwei Chen, William Hardy, Yanfang Ye, and Tao Li. 2015. Analyzing file-to-file relation network in malware detection. In *WISE*. Springer, 415–430.
[5] Lingwei Chen, Shifu Hou, and Yanfang Ye. 2017. SecureDroid: Enhancing Security of Machine Learning-based Detection against Adversarial Android Malware Attacks. In *ACSAC*. ACM, 362–372.
[6] Lingwei Chen, Tao Li, Melih Abdulhayoglu, and Yanfang Ye. 2015. Intelligent malware detection based on file relation graphs. In *ICSC*. IEEE, 85–92.
[7] Lingwei Chen, Yanfang Ye, and Thirimachos Bourlai. 2017. Adversarial Machine Learning in Malware Detection: Arms Race between Evasion Attack and Defense. In *EISIC*. IEEE, 99–106.
[8] Peng Cui, Shaowei Liu, and Wenwu Zhu. 2017. General Knowledge Embedded Image Representation Learning. In *IEEE Transactions on Multimedia*.
[9] Peng Cui, Xiao Wang, Jian Pei, and Wenwu Zhu. 2017. A Survey on Network Embedding. In *arXiv preprint arXiv:1711.08752*.
[10] CybersecurityVentures. 2017. *Ransomware Damage Report*. https://cybersecurityventures.com/ransomware-damage-report-2017-5-billion/.
[11] Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. 2017. metapath2vec: Scalable representation learning for heterogeneous networks. In *KDD*. ACM, 135–144.
[12] Yujie Fan, Yanfang Ye, and Lifei Chen. 2016. Malicious sequential pattern mining for automatic malware detection. *ESWA* 52 (2016), 16–25.
[13] Yujie Fan, Yiming Zhang, Yanfang Ye, Wanhong Zheng, et al. 2017. Social Media for Opioid Addiction Epidemiology: Automatic Detection of Opioid Addicts from Twitter and Case Studies. In *CIKM*. ACM, 1259–1267.
[14] Tao-yang Fu, Wang-Chien Lee, and Zhen Lei. 2017. HIN2Vec: Explore Meta-paths in Heterogeneous Information Networks for Representation Learning. In *CIKM*. ACM, 1797–1806.
[15] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *KDD*. ACM, 855–864.
[16] Jiawei Han, Yizhou Sun, Xifeng Yan, and Philip S Yu. 2010. Mining knowledge from databases: an information network analysis approach. In *SIGMOD*. ACM, 1251–1252.
[17] Peter D Hoff, Adrian E Raftery, and Mark S Handcock. 2002. Latent space approaches to social network analysis. *J. Amer. Statist. Assoc.* 97, 460 (2002), 1090–1098.
[18] 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.
[19] Ilse CF Ipsen and Carl D Meyer. 1995. The angle between complementary subspaces. *Amer. Math. Monthly* (1995), 904–911.
[20] Jeremy Z Kolter and Marcus A Maloof. 2004. Learning to detect malicious executables in the wild. In *KDD*. ACM, 470–478.
[21] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).
[22] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*. 3111–3119.
[23] Ming Ni, Tao Li, Qianmu Li, Hong Zhang, and Yanfang Ye. 2016. FindMal: A file-to-file social network based malware detection framework. *Knowledge-Based Systems* 112 (2016), 142–151.
[24] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *KDD*. ACM, 701–710.
[25] Chuan Shi, Yitong Li, Jiawei Zhang, Yizhou Sun, and S Yu Philip. 2017. A survey of heterogeneous information network analysis. *IEEE Transactions on Knowledge and Data Engineering* 29, 1 (2017), 17–37.
[26] Yizhou Sun, Rick Barber, Manish Gupta, Charu C Aggarwal, and Jiawei Han. 2011. Co-author relationship prediction in heterogeneous bibliographic networks. In *ASONAM*. IEEE, 121–128.
[27] Yizhou Sun and Jiawei Han. 2012. Mining heterogeneous information networks: principles and methodologies. *DMKD* 3, 2 (2012), 1–159.
[28] 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. *VLDB Endowment* 4, 11 (2011), 992–1003.
[29] Symantec. 2016. *2016 Internet Security Threat Report*. https://www.symantec.com/secu-\rity-center/threat-report.
[30] Acar Tamersoy, Kevin Roundy, and Duen Horng Chau. 2014. Guilt by association: large scale malware detection by mining file-relation graphs. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 1524–1533.
[31] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *WWW*. International World Wide Web Conferences Steering Committee, 1067–1077.
[32] Andrei Venzhega, Polina Zhinalieva, and Nikolay Suboch. 2013. Graph-based malware distributors detection. In *WWW*. ACM, 1141–1144.
[33] Chenguang Wang, Yangqiu Song, Haoran Li, and Jiawei Zhang. 2016. Text Classification with Heterogeneous Information Network Kernels. In *AAAI*. 2130–2136.
[34] Chenguang Wang, Yangqiu Song, Haoran Li, Ming Zhang, and Jiawei Han. 2015. Knowsim: A document similarity measure on structured heterogeneous information networks. In *ICDM*. IEEE, 1015–1020.
[35] Daixin Wang, Peng Cui, and Wenwu Zhu. 2016. Structural Deep Network Embedding. In *ACM SIGKDD*.
[36] Tobias Wüchner, Martín Ochoa, and Alexander Pretschner. 2014. Malware detection with quantitative data flow graphs. In *ASIACCS*. ACM, 271–282.
[37] Shuicheng Yan, Dong Xu, Benyu Zhang, Hong-Jiang Zhang, Qiang Yang, and Stephen Lin. 2007. Graph embedding and extensions: A general framework for dimensionality reduction. *TPAMI* 29, 1 (2007), 40–51.
[38] 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.
[39] Yanfang Ye, Lifei Chen, Dingding Wang, Tao Li, Qingshan Jiang, and Min Zhao. 2009. SBMDS: an interpretable string based malware detection system using SVM ensemble with bagging. *Journal in computer virology* 5, 4 (2009), 283.
[40] Yanfang Ye, Tao Li, Donald Adjeroh, and S Sitharama Iyengar. 2017. A survey on malware detection using data mining techniques. *CSUR* 50, 3 (2017), 41.
[41] Yanfang Ye, Tao Li, Yong Chen, and Qingshan Jiang. 2010. Automatic malware categorization using cluster ensemble. In *KDD*. ACM, 95–104.
[42] Yanfang Ye, Tao Li, Qingshan Jiang, Zhixue Han, and Li Wan. 2009. Intelligent file scoring system for malware detection from the gray list. In *KDD*. ACM, 1385–1394.
[43] Yanfang Ye, Tao Li, Qingshan Jiang, and Youyu Wang. 2010. CIMDS: adapting postprocessing techniques of associative classification for malware detection. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 40, 3 (2010), 298–307.
[44] Yanfang Ye, Tao Li, Shenghuo Zhu, Weiwei Zhuang, Egemen Tas, Umesh Gupta, and Melih Abdulhayoglu. 2011. Combining file content and file relations for cloud based malware detection. In *KDD*. ACM, 222–230.
[45] Yanfang Ye, Dingding Wang, Tao Li, and Dongyi Ye. 2007. IMDS: Intelligent malware detection system. In *KDD*. ACM, 1043–1047.
[46] Yanfang Ye, Dingding Wang, Tao Li, Dongyi Ye, and Qingshan Jiang. 2008. An intelligent PE-malware detection system based on association mining. *Journal in computer virology* 4, 4 (2008), 323–334.
[47] Xuchao Zhang, Liang Zhao, Arnold P Boedihardjo, and Chang-Tien Lu. 2017. Online and Distributed Robust Regressions under Adversarial Data Corruption. In *ICDM*. IEEE, 625–634.
[48] Huan Zhao, Quanming Yao, Jianda Li, Yangqiu Song, and Dik Lun Lee. 2017. Meta-graph based recommendation fusion over heterogeneous information networks. In *KDD*. ACM, 635–644.
[49] Liang Zhao, Feng Chen, Chang-Tien Lu, and Naren Ramakrishnan. 2016. Multi-resolution spatial event forecasting in social media. In *ICDM*. IEEE, 689–698.
[50] Liang Zhao, Jiangzhuo Chen, Feng Chen, Wei Wang, Chang-Tien Lu, and Naren Ramakrishnan. 2015. Simnest: Social media nested epidemic simulation via online semi-supervised deep learning. In *ICDM*. IEEE, 639–648.
[51] Liang Zhao, Ting Hua, Chang-Tien Lu, and Ray Chen. 2016. A topic-focused trust model for Twitter. *Computer Communications* 76 (2016), 1–11.
[52] Liang Zhao, Junxiang Wang, and Xiaojie Guo. 2018. Distant-supervision of heterogeneous multitask learning for social event forecasting with multilingual indicators. In *AAAI*. 4498–4505.
[53] Guido Zuccon, Leif A Azzopardi, and CJ Van Rijsbergen. 2009. Semantic spaces: Measuring the distance between different subspaces. In *International Symposium on Quantum Interaction*. Springer, 225–236.