IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received August 12, 2019, accepted October 15, 2019, date of publication October 18, 2019, date of current version November 11, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2948374

Malware Detection via Extended Label Propagation Through Graph Inference

YITU FU[1] AND JU XU[2]
[1]College of Information Science and Engineering, Northeastern University, Shenyang 110819, China
[2]School of Economics and Management, Tongji University, Shanghai 201804, China

Corresponding author: Ju Xu (ju.xu@outlook.com)

ABSTRACT In this paper, we model the malware detection problem as a graph inference problem, and develop a novel belief propagation approach within a semi-supervised learning scheme that fully makes use of files' and hosts' connections to detect malware. Specifically, with network download data, we build a large graph that depicts files' co-occurrence and files-hosts relationship. Different from the classical methods that heuristically define edge weights only in the file co-occurrence graph, we develop a new method to integrate homophilic host-file relationship on top of file co-occurrences. Then, by using the linear neighborhood model, we first perform propagations in the subgraph of files to achieve their stabilization, then extend the propagation to the complete file-host graph. To facilitate this propagation procedure, we develop a set of algorithmic tools that extract information for the linear neighborhood model from the link structure of download events. Also, we theoretically show that, under some mild conditions, our propagation method could reveal the actual labels of unlabeled nodes in the complete graph. Finally, we perform a set of experiments that demonstrate the effectiveness of our new method in a variety of contexts on a real-world dataset.

INDEX TERMS Malware detection, machine learning, data mining, semi-supervised learning, graph algorithms.

I. INTRODUCTION

As the number of computers increases dramatically and they ubiquitous access to a high-speed Internet connection, malware, which is also known as malicious software, has spread and infected computers around the world at an unprecedented rate. Protection against malware attacks is very important as the cost of these attacks increase. In particular, the battle against malware is becoming very difficult as attackers develop extremely sophisticated techniques to break traditional security measures. This suggests malware detection solution will face great challenges in the years to come, as they will likely be outpaced by the threats created by malware authors. Analysis based on executable individual files has always been the main means of malicious file evaluation and detection. Currently, this kind of thinking provides an effective platform for malware detection.

Recent years, cloud services are prominent within the private, public and commercial domains. Many of these services are expected to be always on, therefore, security is a critical consideration. In order to remain secured, a cloud needs to possess the ability to react to threats [1]. In 2008, Oberheide *et al.* [2] first proposed anti-virus cloud service at the conference on USENIX security. More and more scholars and security vendors have researched and released malware detection tools and solutions based on this concept [3]– [5]. Generally, the cloud security service follows the steps described below. Cloud service submits executable files which cannot be determined by the local anti-virus engine on the end host, and summarizes files submitted by the anonymous ID of the end host. These files are either manually evaluated by experts in the security team, or automatically evaluated by traditional defense mechanisms. At last, the evaluated results upgrade the security strategy on end hosts to improve the defense. Fig. 1 exhibits an example of the cloud security service.

The method that is independent of the contents of the file is established under the reputation system by analyzing the credibility of the source of the file. Several previous malware detection works investigate how malware propagates in networks and analyze the temporal propagation pattern of executable files [6]– [11]. Although these methods apply the

The associate editor coordinating the review of this manuscript and approving it for publication was Isaac Triguero.

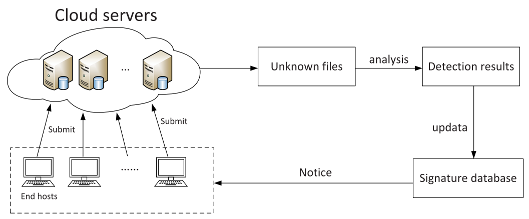

FIGURE 1. **Work flow of cloud security service.**

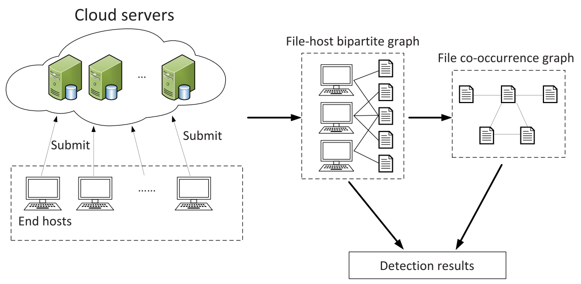

FIGURE 2. **Overview of our approach to malware detection.**

technique to the analysis of characteristics of malware, they ignore the interaction between malicious or benign files and the reputation of end hosts.

There are several previous works providing an effective complement to analytical techniques for malware detection via machine learning algorithms. Chau *et al.* [12] formulate the malware detection problem as a file reputation inference problem over a machine-file bipartite graph. They assume the machine-file relationship can follow homophily. Inferring file goodness through incorporating homophilic machine-file relationships domain knowledge and intuition, and other files' goodness through their influence on associated machines. The focus of Rajab *et al.* [13] is specifically on the reputation of most downloads locally. Their method processes the features sent from the browser and computes a reputation decision informed by the client request and a reputation metric constructed from previous client requests. Rahbarinia *et al.* [14] present an approach for accurate real-time detection of malware download events. This method combine the knowledge derived from historic relationships between machines, files, and UPLs that were triggered by clients, with both system and network properties of each download event. Although the reputation of the container provides evidence of the reputation of the file in it and the reputation of files provides an extremely important basis for these methods, the heuristic or experience-based reputation system makes techniques inflexible to the dramatic increase in new malware. These methods have heavily relied on refining existing signature-based protection models pioneered by the industry. In addition, these machine learning algorithms require an amount of labeled ground truth for training purposes. Researchers discuss methods to incorporate class priors and the predictions of classifiers obtained by supervised learning. The reputation score of files is often, however, very time consuming and expensive to obtain, as the reputation score requires the efforts of human annotators.

Semi-supervised learning is a type of machine-learning technique that is especially useful when a limited amount of labelled data exists for each class [29]. These techniques create a supervised classifier based on labeled data and predict the label for all unlabelled instances. Based on the file-to-file relation network, Chen *et al.* [15] design several robust graph-based features for malware detection and reveal its relationship characteristics. Based on the designed features and two findings, they select the representative samples from the large unknown file collection for labeling and then use the belief propagation algorithm to detect malware. Tamersoy *et al.* [16] leverage locality-sensitive hashing to measure the strength of these inter-file relationships to construct a graph, on which it performs large scale inference by propagating information from the labeled files (as benign or malicious) to the preponderance of unlabeled files. Ni *et al.* [17] construct the file relation graph, k-nearest neighbors are chosen as adjacent nodes for each file node. Their Label propagation algorithm propagates label information from labeled file samples to unlabeled files. The algorithm is used to learn the probability that one unknown file is classified as malicious or benign. However, there are two drawbacks in file co-occurrence based malware detection. One is that their detection methods largely sacrifice the utilization rate of files, because a large number of the unlabeled file is ignored as noise by detection methods during processing. The other is these defenses that ignore the tracking of the latest virus files between hosts are only partially effective, given that they tend to lag behind the latest threats, thus leaving users exposed to new malware infections.

In this paper, we revisit the problem of file co-occurrence and interaction between end host and file by incorporating Semi-supervised learning into the problem formulation. The main challenge here is that there are no partially labeled files available for clusters. Hence the machine learning has to be done in a semi-supervised fashion. To achieve high detection accuracy, we introduce a graph model defined the link structure of the files to discover other files that are likely to be malicious or benign, and then we guide security-related profiling that learns the risk scoring framework for files via user's downloading behavior. This is in contrast with previous works, which attempt to detect malware downloads based primarily on features derived from network traffic [30] [13] or that only consider the relationships between end hosts and files [12].

Fig.2 shows an overview of our approach to malware detection. Due to the cloud security service, real-time files download events are recorded when hosts submit files to the remote server. Our system aims to detect malware by another consideration about who is downloading what files. The importance of an executable file is an inherently subjective matter, which depends on the readers' interests, knowledge, and attitudes [25]. The average files quality experienced by most of the user is higher than the quality of the other average files, which is because the certain special purpose of creating and publishing files results in a large fraction of low quality that the users are unlikely to use [26]. The dependencies are derived from our domain knowledge. User activities on benign hosts result primarily in benign files downloads and occasional unintended malicious files downloads. Hence we may assume that benign hosts are more likely to download benign files than malicious files. Similarly, a benign file's neighbor is more likely to benign host than a malicious host. However, malicious hosts are more likely to visit malicious files as malware tends to be associated with many malicious files. We derive the relationship between files via end hosts' behavior. A good file is likely to be associated with a machine with a good reputation than with a low-reputation one [11]. The idea that the accuracy of file co-occurrence can be significantly improved with a cluster metric tailored to the specific data set is well documented. The method of learning an appropriate cluster metric from a given reputation score or partially reputation score training file has demonstrated to be effective in the literature lately [24].

In the malware detection community, there have been many previous works using belief propagation algorithms or limited neighborhood set approaches. Although those use efficient computational methods, the classification criteria used in most of them are based on just local properties of the billion-node graph. Our method combines knowledge derived from historic relationships between hosts and files, with both files co-occurrence and files-hosts relationship properties of each download events. We prove theoretically that our matrices of graph-based methods can precisely approximate the label propagation framework. Therefore, this approximated matrices can be used as smooth matrices as in standard graph-based malware detection algorithms.

The main contributions of this paper are as follows:

- We present a novel technology that detects malware through large-scale graph inference based on the scalable belief propagation algorithm. It infers every file's reputation, flagging files with a low reputation as malware. Our method leverage large-scale situation awareness about cloud security service-based files download events and casts the problem of detecting new malware downloads as an inference problem over cyclic (files$\rightleftharpoons$files)$\rightarrow$hosts$\rightarrow$files relationship graph.
- Note that the existing approach has been demonstrated to be effective when threats are known in advance, but the methods cannot cope with previously unseen malware, or with large amounts of new malware. Unlike the previous work, our method does not require a significant amount of malicious code and benign contents be identified and labeled beforehand, and thus allows us to more accurately detect future malware downloads.
- We show that the final classification result of our method is much more effective and usable for malware detection at any time, and for unknown malware detection compared to other methods mentioned.

II. METHODS

Our goal is to propose an approach that scales hundreds of millions of users and protects them from downloading malware while at the same time limiting the impact on their privacy. We focus on two aspects, file co-occurrence and homophilic host-file relationship, based on a semi-supervised graph-then-cluster method. In several earlier work, researchers demonstrated that a semi-supervised graph-then-cluster method can produce a reliable classification model from small amounts of labeled data [18]– [22]. In this section, we propose an improvement of the method proposed in earlier studies and we also provide the theoretical justifications of the feasibility of our method.

A. FILE CO-OCCURRENCE

We first review the harmonic property under the graph notation. The harmonic property means that the reputation value of files at each point is the average of files at neighboring points. Harmonic property is at the heart of graph-based methods. As in [23], we assume that all of these neighborhoods are linear, that is, each data point can be optimally reconstructed by using a linear combination of its neighbors. In this paper, given the co-occurrence graph of files, we employ label propagation to infer the reputation value of files.

As outlined briefly in this section, the basic framework presented in the previous section can be viewed in several fundamentally different ways and these different viewpoints provide a rich and complementary set of techniques for reasoning about this approach to the semi-supervised learning problem. Imagine a particle walking along the graph, starting from a node, it moves to another node with probability after one step, the walk continues until the particle hits all over the node. To provide some intuition on how most semi-supervised methods works, Wang *et al.* [20] described how the labels propagate through the linear neighborhoods.

In practice, a file f_i has no multiple direct hyperlinks to another file f_j. Fig. 3 represents a example which implies a bipartite graph of files and end hosts, an edge between them indicating the emergence of the file in the end host. We consider two files as co-occurrence if they are downloaded by the same host. Roughly speaking, the individual files that share the same host belong to the same cluster. Under this definition, we build a file co-occurrence graph which is an undirected graph with each vertex representing a distinct file. We add one or more links between file f_i and file f_j if there are one or more times of file f_i and file f_j co-occurrence in the original cluster graph.

We derive a reliable and stable way for malware detection if there are only very few labeled files available. We propose to use the neighborhood information of each file to construct $\boldsymbol{G} = (\boldsymbol{F}, \boldsymbol{W})$ with the files set $\boldsymbol{F} = f_1, f_2, \ldots, f_n$, $\forall f \in \boldsymbol{F}$ can assign a real value of reputation to every file, and the weight

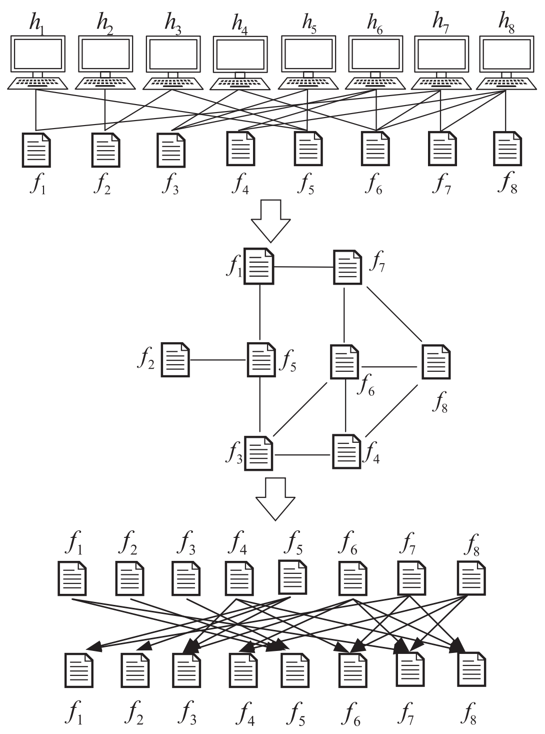

FIGURE 3. An example of the files co-occurrence graph and the labels propagate through the linear neighborhoods.

$w_{ij} \in \boldsymbol{W}$ reflects the similarity between f_i and f_j. We assume the $n \times n$ symmetric weight matrix $\boldsymbol{W}$ on the edges of the graph is given. For example, when $(x_{i1}, x_{i2}, \ldots, x_{im}) \in \mathbb{R}$ is the feature vector for f_i, the weight matrix is

$$w_{ij} = \exp(-\sum_{d=1}^{m} \frac{(x_{id} - x_{jd})^2}{\sigma_d^2}) \tag{1}$$

where the variance $\sigma_1, \ldots, \sigma_m$ are free parameters. File co-occurrence of the function states that the reputation value of a file is the linear combination of the value of its neighbors. After all of the reconstructing weights are computed, we can construct a sparse matrix $\boldsymbol{W}$ by

$$\boldsymbol{W}(i,j) = w_{ij} \tag{2}$$

Given in matrix diag(d_i) is the diagonal matrix with entries $d_i = \sum_j w_{ij}$

$$d = \max\left(d_i = \sum\nolimits_j w_{ij}\right) \tag{3}$$

After the graph is constructed, we have to make use of it to predict the labels of the unlabeled vertices. Here, we propose a scheme to the one in *Label Propagation*, which can iteratively propagate the labels of the labeled data to the remaining unlabeled data on the constructed graph. The process of labels propagate property means that the value of f_i at each data point is the average of f_i at neighboring points, Therefore, the label of the reputation of f_i at $m+1$ times propagation steps.

$$f_i^{m+1} = \frac{\alpha}{d}\sum_{j=1}^{n} w_{ij} f_j^m + (1-\alpha)\, t_i \tag{4}$$

where $0 < \alpha < 1$ is the fraction of label information that f_i receives from its neighbors. Let $\boldsymbol{t} = (t_1, t_2, \ldots, t_n)^T$. $\boldsymbol{F}^m = (f_1^m, f_2^m, \ldots, f_n^m)^T$ is the prediction label vector at iteration t, and $\boldsymbol{F}^0 = \boldsymbol{t}$. The equivalent matrix equation is:

$$\boldsymbol{F}^{m+1} = \frac{\alpha}{d}\boldsymbol{W}\boldsymbol{F}^m + (1-\alpha)\,\boldsymbol{t} \tag{5}$$

We will use (5) to update the labels of each data object until convergence. First, we will analyze the convergence property of the iterative formula (5) and propose a one-shot. Specifically, we have the following theorem:

Theorem 1: The sequence $\boldsymbol{F}^m$ *computed by (5) will converge to*

$$\boldsymbol{F}^m = (1-\alpha)\left(I - \frac{\alpha}{d}\boldsymbol{W}\right)^{-1}\boldsymbol{F}^0 \tag{6}$$

when $m \to 0$ and $\boldsymbol{F}^0$ is the initial condition.

Proof: By (5), we have

$$\boldsymbol{F}^m = \left(\frac{\alpha}{d}\boldsymbol{W}\right)^{m-1}\boldsymbol{F}^0 + (1-\alpha)\sum_{i=0}^{m-1}\left(\frac{\alpha}{d}\boldsymbol{W}\right)^i \boldsymbol{F}^0 \tag{7}$$

Since $w_{ij} \geq 0$, $d_i = \sum_j w_{ij}$ and $d = \max\left(d_i = \sum_j w_{ij}\right)$. We will construct a other sparse marix $\boldsymbol{M}$:

$$\boldsymbol{M}(i,j) = (m_{ij}) = \left(\frac{w_{ij}}{d_i}\right) \geq \left(\frac{w_{ij}}{d}\right) = \frac{1}{d}\boldsymbol{W}(i,j) \tag{8}$$

Since $m_{ij} \geq 0$ and $\sum_j m_{ij} = 1$, from the theorem of *Perron-Frobenius* [27], we know that spectral redius of $\boldsymbol{M}$ or $\rho(\boldsymbol{M}) \leq 1$, and $0 < \alpha < 1$, thus,

$$\lim_{m\to\infty} (\alpha\boldsymbol{M})^{m-1} = 0 \tag{9}$$

Based on the formula (8), we can easily derive the following theorem:

$$\begin{gathered} 0 \leqslant \lim_{m\to\infty}\left(\frac{\alpha}{d}\boldsymbol{W}\right)^{m-1} \leqslant \lim_{m\to\infty}(\alpha\boldsymbol{M})^{m-1} = 0 \\ \lim_{m\to\infty}\sum\nolimits_{i=0}^{m-1}\left(\frac{\alpha}{d}\boldsymbol{W}\right)^i = \left(\boldsymbol{I} - \frac{\alpha}{d}\boldsymbol{W}\right)^{-1} \end{gathered} \tag{10}$$

where $\boldsymbol{I}$ is the identity matrix of order n, Clearly, the sequence $\boldsymbol{F}^m$ will converge to

$$\boldsymbol{F} = \lim_{m\to\infty}\boldsymbol{F}^m = (1-\alpha)\left(\boldsymbol{I} - \frac{\alpha}{d}\boldsymbol{W}\right)^{-1}\boldsymbol{F}^0 \tag{11}$$

which proves a theorem. Hence, we can use $\boldsymbol{F}$ as the classification function, which results in a one-shot method. That is, we can predict the labels of all the data objects in one step.

The common denominator of graph-based methods is to model the whole data set as a graph $\boldsymbol{G} = (\boldsymbol{F}, \boldsymbol{W})$, as we have mentioned in this section. But just studying the relationship between files cannot provide a strong basis for malware detection. In our case, the homophilic host-file relationship based on file co-occurrence is at the heart of the malware detection method.

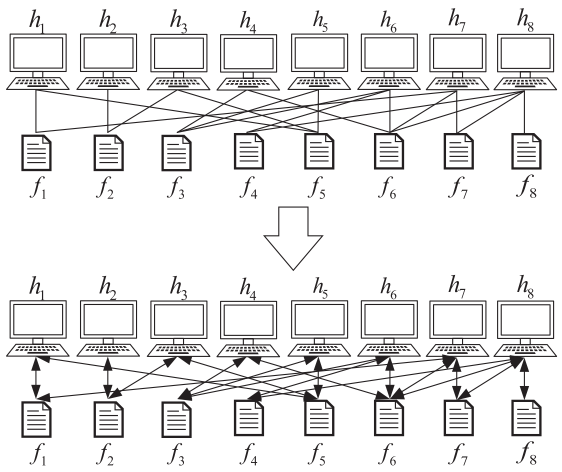

FIGURE 4. An example of the bipartite graph of files and end hosts the labels propagate through the linear neighborhoods.

B. HOMOPHILIC HOST-FILE RELATIONSHIP BASED ON FILE CO-OCCURRENCE

We believe that focusing on the behavior of user downloading files is justified for several reasons: with the gradual upgrade of the anti-virus engine's defense capabilities in recent years, the anti-virus engine gives appropriate prompts when the host downloads files that are not unknown binary by the anti-virus engine. User activities on benign hosts result primarily in benign file accesses and occasional unintended malicious file access. The benign hosts are more likely to visit benign files than malicious files. Similarly, a benign file's neighbor is more likely to be a benign host than a malicious host. Malicious hosts are more likely to visit malicious files as malware tend to contact many malicious files [11]. The fact that homophilic host-file relationships that, good files are more likely to appear on downloading the list of end hosts with a good reputation and bad files are more likely to appear on downloading the list of end hosts with a low reputation.

This section presents the procedure of the labels propagate through the bipartite graph of files and end hosts on a example. Fig. 4 shows an example of the dataset of file submission records obtained from the cloud security service. We construct the bipartite graph of files and end hosts, according to this submission records. Due to the cloud security service, when hosts submit files to the remote server, the timestamp is recorded as well. We consider file-file relationships with little correlation if there is a one-hour interval between the files' submission from the same host. Thus, we split the original whole dataset into multiple datasets according to the time interval that they belong to, and we consider host-file relationships at the same time interval only. We evaluate the performance of malware detection under different time in our experiments as shown in the later subsection. Furthermore, as shown in Fig. 4, we build the union graph of files and end hosts. We conduct edge-weight learning and label prediction on the graph.

Apparently host-file relationship is responsible for creating malware detection method. We assume there are l hosts $h_1, h_2, \ldots, h_l$ which denote the reputation score of the host and m files $f_1, f_2, \ldots, f_m$ which denote the reputation score of the file. We assume an $l \times m$ edges matrix $\boldsymbol{E}$ on the edges of the graph is given. For example, when the ith host download the jth file once or twice, the edges matrix can be $e_{ij} = 1\, or 2$. For our purposes, the matrix $\boldsymbol{E}$ fully specifies the data manifold structure. The reputation score of each host is the summation of the reputation score of files which is downloaded by the host: $h_i = \sum_j e_{ij} f_j$. On the contrary, we use the same method to get reputation score of ith file: $f_i = \sum_j e_{ji} h_j$. Above on, host-file relationships becomes: $\boldsymbol{H} = \boldsymbol{EF}$, $\boldsymbol{F} = \boldsymbol{E}^T \boldsymbol{H}$.

$$\begin{aligned} \boldsymbol{H}^{m+1} &= \boldsymbol{E}\boldsymbol{F}^m \\ \boldsymbol{F}^{m+1} &= \boldsymbol{E}^T \boldsymbol{H}^m \end{aligned} \tag{12}$$

Given the file co-occurrence mentioned above, we can develop the method to integrate homophilic host-file relationship on top of file co-occurrences, meanwhile, the labels of files are propagating through the linear neighborhoods. In fact, in each propagation step, we let each data file absorb a fraction of label information from its neighbors and then exchange some label information of its hosts.

$$\begin{aligned} \boldsymbol{H}^{m+1} &= \boldsymbol{E}\widehat{\boldsymbol{W}}\boldsymbol{F}^m \\ \boldsymbol{F}^{m+1} &= \boldsymbol{E}^T \boldsymbol{H}^m \end{aligned} \tag{13}$$

It is straightforward to associate File co-occurrence assumption with $Theorem 1$ mentioned in last subsection. We imagine that the sequence $\{\boldsymbol{F}^m\}$ propagates the fraction of label information to their linear neighbors until convergence, and then the relabeled sequence $\{\boldsymbol{F}^m\}$ continues to propagate label information to their hosts who download them. We can consider the sequence $\{\boldsymbol{F}^m\}$ in the second formula of (13) as the initial condition of the first formula at each time. Using the conclusion of $Theorem 1$, we can rewrite our iteration equation.

Theorem 2: The prediction results of our method can be derived from the reconstructive framework:

$$\begin{aligned} \boldsymbol{H}^{m+1} &= \boldsymbol{E}\left[(1-\alpha)\left(I - \frac{\alpha}{d}\boldsymbol{W}\right)^{-1}\right]\boldsymbol{F}^m \\ \boldsymbol{F}^{m+1} &= \boldsymbol{E}^T \boldsymbol{H}^m \end{aligned} \tag{14}$$

where $(\boldsymbol{W})_{ij}$ is the weight matrix by solving (1) for the reconstruction weights of each data object from its neighbors. Fig.5 shows the basic idea of the procedure described above, which underscores the file co-occurring as well as the host-file relationship. We use (14) to update the labels of each data object until convergence. Convergence means that the predicted labels of the data will not change in several successive iterations. The convergence analysis of our algorithm will be presented in the next subsection. After the graph has been constructed, we make use of it to predict the labels of the unlabeled vertices.

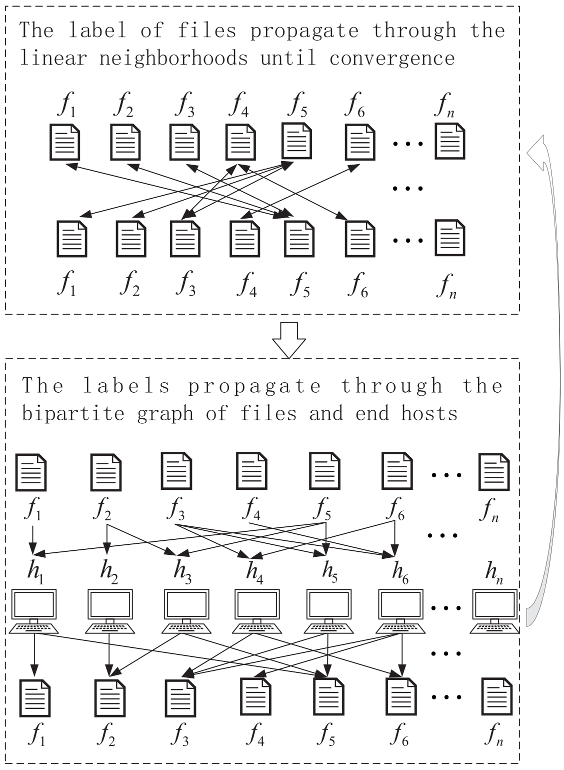
The label of files propagate through the linear neighborhoods until convergence
f_1 f_2 f_3 f_4 f_5 f_6 f_n
The labels propagate through the bipartite graph of files and end hosts
h_1 h_2 h_3 h_4 h_5 h_6 h_n

FIGURE 5. The example representation of approach overview.

```
Algorithm 1 Framework of Host-File Relationship Based on File Co-Occurrence
Input: $F^0 = (f_1, f_2, \dots, f_n) \in \mathbb{R}$ are initial label of files;
Output: The labels of all the data files.
    Construct the neighborhood graph of files by solving (1).
    From the weight matrix $(W)_{ij}=w_{ij}$;
    Construct edges matrix between files and hosts: $E$:
    Compute the parameter $d = \max(d_i = \sum_j w_{ij})$
    repeat
        $H^{m+1} = E\left[(1-\alpha)\left(I - \frac{\alpha}{d}W\right)^{-1}\right]F^m$;
        $H^{m+1} = H^{m+1} / \|H^{m+1}\|_2$;
        Update $F^{m+1} = E^T H^{m+1}$;
    until convergence
```

C. CONVERGENCE ANALYSIS

In this section, we will prove the iterative process of (14).

Proof: By (14) and the initial condition that F^0 is given. W is an $N \times N$ symmetrical matrix with $W(i,j) = w_{ij}$. One easily verifies that $(1-\alpha)\left(I - \frac{\alpha}{d}W\right)^{-1}$ is also $N \times N$ symmetrical matrix. We will do so by transforming $(1-\alpha)\left(I - \frac{\alpha}{d}W\right)^{-1}$ into a standard symmetrical matrix. Rewrite (14) as

$$\begin{aligned} H^{m+1} &= E\widehat{W}F^m \\ F^{m+1} &= E^T H^m \end{aligned} \tag{15}$$

where $\widehat{W} = (1-\alpha)\left(I - \frac{\alpha}{d}W\right)^{-1}$. Based on the above lemma, we can easily derive the following

$$H^{m+1} = (E\widehat{W}E^T)^m E\widehat{W}F^0 \tag{16}$$

$(E\widehat{W}E^T)^T = E\widehat{W}E^T$ indicates that $E\widehat{W}E^T$ is a symmetric matrix. A standard result of linear algebra states that if $E\widehat{W}E^T$ is a symmetric matrix, there will be orthonormal $X = (x_1, \dots, x_n)$ satisfies $X^{-1}(E\widehat{W}E^T)X = diag(\lambda_1 \dots \lambda_n)$. Assume that $|\lambda_1| > |\lambda_2| \geq \dots \geq |\lambda_n|$, $E\widehat{W}F^0 = \alpha_1 x_1 + \alpha_2 x_2 + \dots + \alpha_n x_n$, and $(\alpha_1 \neq 0)$, then

$$\begin{aligned} H^{m+1} &= (E\widehat{W}E^T)^m E\widehat{W}F^0 \\ &= \alpha_1 \lambda_1^m \left(x_1 + \frac{\alpha_2}{\alpha_1}\left(\frac{\lambda_2}{\lambda_1}\right)^m x_2 + \dots + \frac{\alpha_n}{\alpha_1}\left(\frac{\lambda_n}{\lambda_1}\right)^m x_n\right) \end{aligned} \tag{17}$$

we conclude that

$$dist(H^{m+1}, \alpha_1 \lambda_1^m x_1) = o\left(\left|\frac{\lambda_i}{\lambda_1}\right|^m\right) \tag{18}$$

$$H^{m+1} \big/ \left\|H^{m+1}\right\|_2 = x_1 \tag{19}$$

then renew H^{m+1}

$$H^{m+1} = H^{m+1} \big/ \left\|H^{m+1}\right\|_2 \tag{20}$$

Clearly, the normalizing vector of H^{m+1} will converge to x_1, Hence, we can use $F^{m+1} = E^T H^{m+1}$ to get the classification function and we can predict the labels of all the data objects in several successive iterations.

EXPERIMENTS

In this section, we conduct numerical experiments to evaluate the performance of the method based on homophilic host-file relationship and file co-occurrence in a semi-supervised manner for malware detection. The experiments are conducted by utilizing a real-world dataset from the cloud security service of RiSing.

D. DIGIT RECOGNITION

We focus on the problem of classifying files. The one day-data set we adopt is from RiSing, a Chinese security company. In China, a large fraction of users join the cloud security programs provided by RiSing. According to the dataset from the real world, we originally get a total of 700,000 pieces of data of 24 hours. Each piece of the data is thus represented by an 18-dimensional vector which includes ID addresses of a host and a file, the reputation score of the file, and the other 15 features of the file. It means that the dataset record all the details of user downloads. 8060 end hosts submitted 134410 executable files. Among this set of data, 29389 files are confirmed as benign and 25041 files are confirmed as malware, and that can provide a basis for our experimental study. The reputation scores of the rest of the files have not been confirmed yet.

This paper analyzes the real data of the cloud security service. One interesting phenomenon we have noticed from the real dataset is that the quantity of files downloaded during the day is larger than that at night. On the contrary, the ratio of malicious files that downloaded at night is 63.94 percent among all malicious files whose reputations have been confirmed. We grouped all the files and 8060 hosts, using the host-file relationship that is part of our method. After the analysis of data, we find that several individual hosts who download files at daytime share several good files, but this phenomenon is not obvious at night. According to this result, we can imagine that most people punctually and regularly download benign files for working, learning or the other else in the daytime. But a few of end hosts ignore security company recommendations download more amount of files which may be malicious than the other end hosts. This fact will gradually become apparent over time, especially at midnight. The situation is more noticeable in the daytime than at night. In other words, the architecture of host-file and file-file never stop change while the time keeps going. According to our case study, we will also focus on the problem of detection malware during day and night, and discussion of this subject will be in a later subsection.

E. COMPARED METHODS AND SETUP

We consider the binary problem of classifying files ''benign'' vs. ''malware''. We report average accuracy of the following methods on unlabeled data: *PageRank* [26], *TrustRank* [31] and *HITS* [32]. *PageRank*, *TrustRank* and *HITS* are used simply as baselines. *PageRank* has been used to evaluate the importance of crawled pages over the web. A relevant interpretation of this rank is the probability of staying or entering a webpage during a random walk. Previous studies [33] based on *PageRank* motive the implementation of such types of algorithms. *HITS* evaluates the importance of webpages using characteristics: Authority and Hub. The algorithm is based on the following observation: a good hub points to good authorities, and a good authority is pointed to by good hubs, similar to the homophilic host-file relationship which is one of our focus. However, previous studies [34] [35] widely use the implementation of *HITS* on the larger graph that covers several queries. We choose these classifiers as the baselines because of their popularity and empirical success. In our basic approach, the solution is also based on the structure of the data manifold, which is derived from data features. Our approach of semi-supervised learning is based on *cluster assumption* [36], which states that two points are likely to have the same class label if there is a path connecting them. For comparison, we also provide the classification results achieved by Zhu et al.'s *Gaussian fields* approach [19], Wang et al.'s *Linear Neighborhood Propagation(LNP)* [20] and Zhou et al.'s *consistency* method [37], that can propagate the labels from the labeled points to the whole data set.

We evaluate the reputation score of files on the real host-file and file-file co-occurrence graph for each hour. According to the previous subsection, the total number of the dataset we collected, including hosts files and feathers of files, is 700,000. And then, we split the 700,000 pieces of data

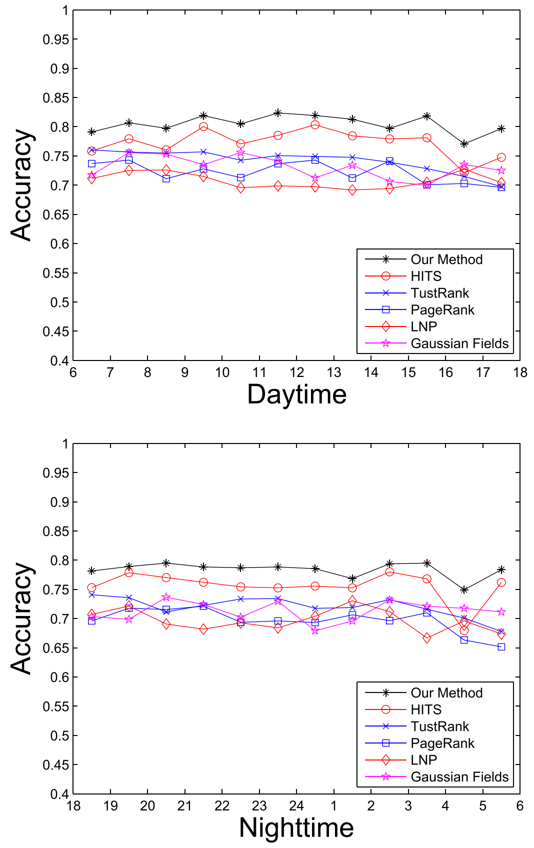

FIGURE 6. The results of malicious files detection experiments during 24 hours.

into 24 equal datasets, based on the time. To assure diversity, we adopt the following sampling method. We generate the list of sites in decreasing order of time, and we segment each of 24 equal datasets into 10 buckets. Each of the buckets contains several labeled datasets that can provide ground truth for testing.

F. DIFFERENT TIME

We first compare the performance of our method and baselines between day and night. In the subsection of digit recognition, we describe the architectures of host-file and file-file never stop change. We benchmark the performance of all the methods we mentioned with the same dataset of each hour. Fig.6 and Fig.7 provide a side-by-side comparison of our method and others with respect to the accuracy of benign and malicious file detection in each hour during a day.

Remember, a few hosts may ignore any pieces of advice on anti-virus engines, nor do they explicitly refuse to download malicious files. The purpose of customers who download too many virus files is essentially different from the purpose of those who download normal files. In other words, host-malicious file relationship and host-benign files relationship are different. Fig.6 and Fig.7 respectively display the results of malicious and benign files detection experiments

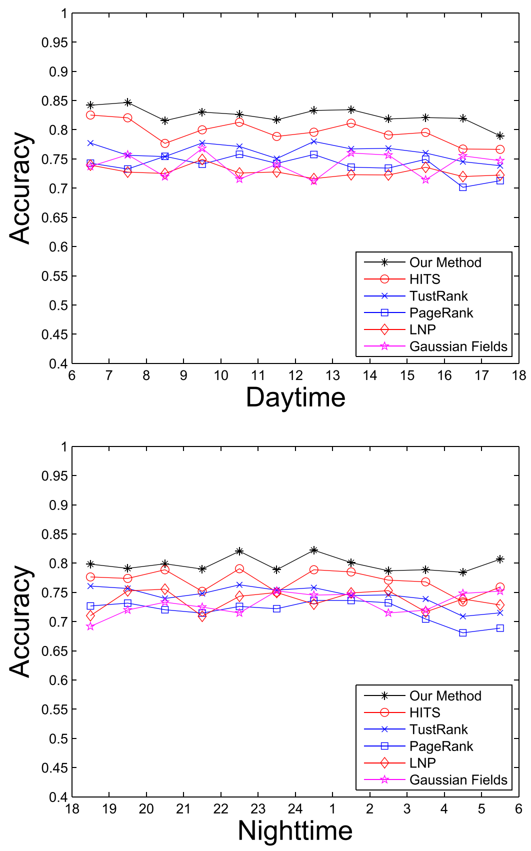

FIGURE 7. The results of benign files detection experiments during 24 hours.

for 24 hours. The horizontal axis shows the accuracy of different algorithms. The recognition accuracies with 400 and 800 pieces of labeled data randomly selected for training and remaining data points for testing. In both figures the vertical axis represents hours. Because we have segmented each of 24 equal datasets into 10 buckets, the ordinate is the total recognition accuracy value averaged over 10 independent runs.

From these figures, we see that our method is a reasonable spam detection tool, that is, even when we test all the methods at any time, it can still get a high recognition accuracy. We believe this is because the dataset of different digits reside on different sub-manifolds, whereas the dataset of the same digit may reside on the same submanifold. Our method can effectively reveal these manifold structures which are reflected by file co-occurrence and interaction between end host and file. As shown in figures, during working hours, the recognition accuracy becomes higher since the bridge points that connect the same digit sub-manifolds are constructed. In other words, most end hosts usually download a lot of execution files for their work during working hours. It is straightforward to improve the cluster assumption of files-file and host-file with these execution files are in common use at daytime since the central idea of our method is to construct two global coordinate systems for the dataset.

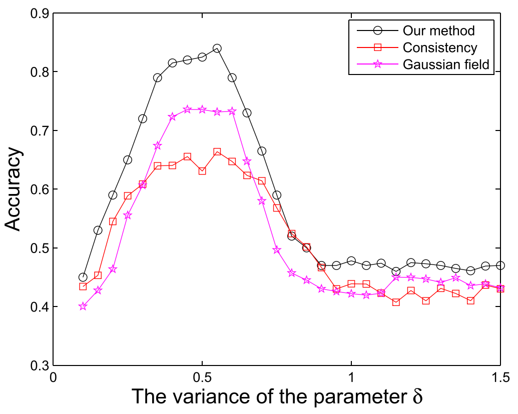

FIGURE 8. The sensitivity of semi-supervised methods to the choice of δ.

An explanation for why our approach to semi-supervised learning at daytime is so effective on the dataset may lie in the common use of executive files. file f_1 and file f_2 coexist in a user's download list, f_3 and f_2 coexist in others, and so on. Thus, although documents far apart in the thread may be quite different, they are linked by edges in the graphical representation of the data, and these links are exploited by the learning algorithm.

G. VARIANCE WEIGHT MATRIX W

In this experiment, we address the task of the different parameters using the 520268 pieces of data during the daytime. Fig.8 offers another view on the performance of our method, *Gaussian fields*, and *Consistency*. It introduces the property of edges which correspond to pairs of co-occurred files. The edge weight quantifies the correlation of the label of co-occurred files. Our edge-weight experiment aims to estimate the parameters δ to better characterize the edge weight in the prediction phase. The horizontal axis of Fig.8 marks the recognition accuracy at the daytime and night, respectively. The vertical axis of the figures corresponds to the value of δ. In the figures, different lines represent the results of different methods, and the values on the line represent the average classification accuracies over 10 independent runs. We assume that the files downloaded by the same host belong to the same cluster and are near to each other. We use this assumption to find the k nearest neighbors when constructing the neighborhood graph in different methods. We use the following function on inner product distance.

$$d\left(x_i, x_j\right) = 1 - \frac{x_i^T x_j}{\|x_i\| \left\|x_j\right\|} \tag{21}$$

where x_i and x_j are respectively document vectors of f_i and f_j. Two files f_i, f_j are connected by an edge if f_i is among f_j's near neighbors or if f_j is among f_i's near neighbors. For Zhou et al.'s *Consistency* and Zhu et al.'s *Gaussian fields*, the affinity matrices were all computed by

$$(W)_{ij} = \exp\left(-\frac{1}{\delta^2}\left(1 - \frac{x_i^T x_j}{\|x_i\| \left\|x_j\right\|}\right)\right) \tag{22}$$

We use the same parameter δ as in Zhou et al.'s *Consistency* and Zhu et al.'s *Gaussian fields* methods is set. We test the parameter stability in all methods mentioned by us at this subsection. A small δ indicates that the weight is more sensitive to variations in the dimension. The experimental results are shown in Fig.8. We find that the methods are very unstable in this experiment since it can only achieve a high classification accuracy when δ falls into a very small range. But our method performs better, and it can hold a little higher classification accuracy as long as δ selects any value between 0.1 and 1.5. We show the sensitivity of our method to the choice of δ, from which we can see that there is a sudden increase when δ changes from 0.1 to 0.3, which indicates that incorporating the smoothness term can greatly improve the effectiveness of our method. Our method is relatively less sensitive to the number of unlabeled points than the other semi-supervised method since slighter changes in the performance of our method are observed as the number of unlabeled points. Besides, the performances of all the methods do not exhibit a significant difference in the variance of the training ratio. According to convergence analysis, the performance of our method has no relationship with the training ratio. It is observed that just the effectiveness of the training ratio can help the other semi-supervised methods to get a high recognition accuracy.

To demonstrate the effects of estimating W on different methods, there is only a single parameter of δ to learn. As noted earlier, when δ falls into a small range (between 0.1 and 0.3), we may find all methods we mentioned are unstable in this experiment. When δ keeps growing towards infinity (we use $\delta = 1.5$ as maximum), δ stabilizes at 0.6, and the classification accuracies of our method reach the best. We can imagine that δ towards infinity is legitimate, it means that the learning algorithm has identified the x-direction as irrelevant, based on all datasets. Under such conditions, the results of all methods are usually undesirable. If we adopt a separate δ for each method, parameter learning is more dramatic. The results of our method achieve higher accuracy than either method, suggesting there is complementary information which is the relationship of host-file used by ours. The figure again shows that our method effectively reflects most of the spam from among each dataset.

H. DIFFERENT DATA SIZE

In the last subsection, we consider the binary problem of classifying with the variance of the Gaussian function. We report the average accuracy of the semi-supervised learning algorithms that we mentioned on unlabeled data. We notice that our method is very sensitive to different values of δ. In this case study, we will focus on the classification accuracies with different date sizes for our method under the situation of variance δ.

Fig.9 illustrates the induction results of our method on the different sizes of the testing dataset. We fixed the size of the dataset to be 1,000, 10,000 and 50,000. 1,000 data points are not enough for discovering the structure of

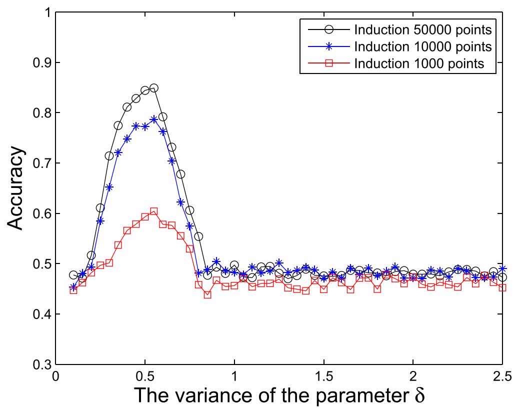

FIGURE 9. The sensitivity of our method to the choice of δ on the different dataset.

file-file and host-file, so the malicious files and benign files detection accuracies are poor. Moreover, 50,000 points are effective for describing this structure in these malicious and benign file detection accuracies that can approximate the accuracies achieved by the standard our method, which uses the whole data set for detection. The induction results are shown in Fig.9, where we also use varied δ to find the classification accuracies when we use a dataset containing the different sizes of points for testing. From Fig.9 we can draw the same conclusion as in the experiments on variance parameter when the dataset can provide sufficient data. That is, when δ changes from 0.4 to 0.7, our induction method can produce results sufficiently approximating the stable results.

Fig.9 exhibits the sensitivity of our method to the choice of δ on the different data size, where the abscissa represents the value of δ, and the ordinate represents the average classification accuracies of our method. In the figure, different lines represent the results with different size of the dataset, and the values on the line represent the average malicious and benign files detection accuracies.

The result for the parameter δ is shown in Fig.9. From the analysis in Theorem 3, we know that the role of δ as the variance of the Gaussian function is to trade off the prediction loss and smoothness. The figure also verifies that we should consider both the prediction loss and smoothness in classification since the performance of our method could be worse when δ becomes either too large or too small.

I. DETECTION OF NEW MALWARE

Both malware variants and new malware are growing rapidly on the Internet. The above experimental setting provides a way of validating the performance of models on detecting malware variants. To validate the performance on detecting new malware, we split the dataset into the training set and testing set as a fresh new dataset according to the temporal information. We treat the dataset of the first a few hours the training set, and the rest dataset as the testing set.

As can be observed from Fig.10, our model on file-file co-occurrence and host-file relationship graphs achieve

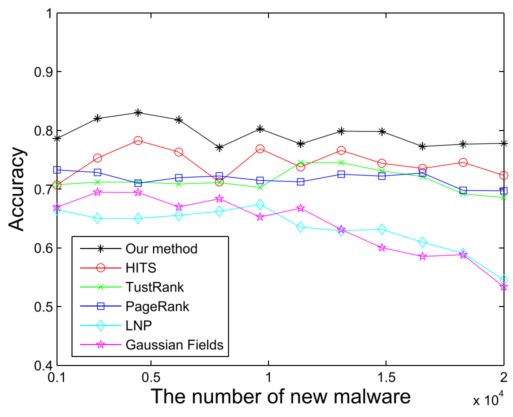

FIGURE 10. **Detection results of models in detecting new malware.**

better performance than other methods on the original file co-occurrence graph. Besides, considering the performance shown in Fig.10, we choose $\delta = 0.65$, since models with this setting achieve better performance than with other settings overall. Fig.10 illustrates the detection results of different methods.

As shown in Fig.10, our model with the error weight in the form of reciprocal of degree outperforms other models in detecting new malware in all experiments, which demonstrates the feasibility of the proposed file-file co-occurrence and host-file relationship graphs. Compared with the results of the experiments mentioned above, the results on new malware are worse, but as the training data continues to increase, the results are getting better. One possible reason is that the experimental dataset does cover all file-file co-occurrence and host-file relationship among the target files. Thus, splitting the training and brand new dataset by time in our experimental dataset leaves many isolated groups of brand new files in the file co-occurrence graph. In contrast, as the testing data continues to increase, splitting the training and brand new dataset leaves a larger connected co-occurrence graph, which results in better performance via the label propagation on the graph.

III. CONCLUSION

In this paper, we revisit the problem of file co-occurrence and interaction between host end and file by incorporating Semi-supervised Learning into the problem formulation. We propose a novel semi-supervised learning algorithm that can discover the structure of the whole data set by synthesizing the linear neighborhood around each data object. We also analyze theoretically that the resulting data labels can be sufficiently smooth with respect to the data structure. Promising experimental results have been presented for malicious files detection, demonstrating that the framework has the potential to effectively exploit the structure of file-file and file-end host to improve malicious files detection accuracy. Finally, we provide experiments to show the effectiveness of our method, from which we find that the method also has specific parameter stability. Meanwhile, as the number of features is rather small in this paper, the time consumption is acceptable to derive the solution analytically. In our future work, we will focus on the theoretical analysis and accelerating issues of our algorithm.

REFERENCES

[1] M. R. Watson, N.-Ul-H. Shirazi, A. K. Marnerides, A. Mauthe, and D. Hutchison, “Malware detection in cloud computing infrastructures,” *IEEE Trans. Dependable Secure Comput.*, vol. 13, no. 2, pp. 192–205, Mar./Apr. 2015.

[2] J. Oberheide, K. Veeraraghavan, E. Cooke, J. Flinn, and F. Jahanian, “Virtualized in-cloud security services for mobile devices,” in *Proc. ACM 1st Workshop Virtualization Mobile Comput.*, 2008, pp. 31–35.

[3] C. Jarabek, D. Barrera, and J. Aycock, “ThinAV: Truly lightweight mobile cloud-based anti-malware,” in *Proc. ACM 28th Annu. Comput. Secur. Appl. Conf.*, 2012, pp. 209–218.

[4] J. Oberheide, E. Cooke, and F. Jahanian, “Rethinking antivirus: Executable analysis in the network cloud,” in *Proc. HotSec*, 2007, pp. 1–5.

[5] J. Oberheide, E. Cooke, and F. Jahanian, “CloudAV: N-version antivirus in the network cloud,” in *Proc. USENIX Secur. Symp.*, 2008, pp. 91–106.

[6] J. Caballero, C. Grier, C. Kreibich, and V. Paxson, “Measuring pay-per-install: The commoditization of malware distribution,” in *Proc. Usenix Secur. Symp.*, 2011, pp. 1–16.

[7] B. J. Kwon, J. Mondal, J. Jang, L. Bilge, and T. Dumitraş, “The dropper effect: Insights into malware distribution with downloader graph analytics,” in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur.*, 2015, pp. 1118–1129.

[8] G. Mezzour, M. M. Carley, and L. R. Carley, “An empirical study of global malware encounters,” in *Proc. ACM Symp. Bootcamp Sci. Secur.*, 2015, Art. no. 8.

[9] S. Yu, G. Gu, A. Barnawi, S. Guo, and I. Stojmenovic , “Malware propagation in large-scale networks,” *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 1, pp. 170–179, Jan. 2015.

[10] E. E. Papalexakis, T. Dumitras, D. H. Chau, B. A. Prakash, and C. Faloutsos, “Spatio-temporal mining of software adoption & penetration,” in *Proc. IEEE/ACM Int. Conf. Adv. Social Netw. Anal. Mining (ASONAM)*, Aug. 2013, pp. 878–885.

[11] P. K. Manadhata, S. Yadav, P. Rao, and W. Horne, “Detecting malicious domains via graph inference,” in *Proc. Eur. Symp. Res. Comput. Secur.* Cham, Switzerland: Springer, 2014, pp. 1–18.

[12] D. H. P. Chau, C. Nachenberg, J. Wilhelm, A. Wright, and C. Faloutsos, “Polonium: Tera-scale graph mining and inference for malware detection,” in *Proc. SIAM Int. Conf. Data Mining*. Philadelphia, PA, USA: SIAM, 2011, pp. 131–142.

[13] M. A. Rajab, L. Ballard, N. Lutz, P. Mavrommatis, and N. Provos, “CAMP: Content-agnostic malware protection,” Tech. Rep., 2013.

[14] B. Rahbarinia, M. Balduzzi, and R. Perdisci, “Real-time detection of malware downloads via large-scale URL-> file-> machine graph mining,” in *Proc. 11th ACM Asia Conf. Comput. Commun. Secur.*, 2016, pp. 783–794.

[15] L. Chen, W. Hardy, Y. Ye, and T. Li, “Analyzing file-to-file relation network in malware detection,” in *Proc. Int. Conf. Web Inf. Syst. Eng.* Cham, Switzerland: Springer, 2015, pp. 415–430.

[16] A. Tamersoy, K. Roundy, and D. H. Chau, “Guilt by association: Large scale malware detection by mining file-relation graphs,” in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2014, pp. 1524–1533.

[17] M. Ni, Q. Li, H. Zhang, T. Li, and J. Hou, “File relation graph based malware detection using label propagation,” in *Proc. Int. Conf. Web Inf. Syst. Eng.* Cham, Switzerland: Springer, 2015, pp. 164–176.

[18] M. PAnkerst and M. H. P. R. T. J. Breunig Kriegel Ng Sander, “Ordering points to identify the clustering structure,” in *Proc. ACM SIGMOD*, vol. 99., 2008.

[19] X. Zhu, Z. Ghahramani, and J. D. Lafferty, “Semi-supervised learning using Gaussian fields and harmonic functions,” in *Proc. 20th Int. Conf. Mach. Learn. (ICML)*, 2003, pp. 912–919.

[20] F. Wang and C. Zhang, “Label propagation through linear neighborhoods,” *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 1, pp. 55–67, Jan. 2007.

[21] M. Zhao, T. W. S. Chow, Z. Zhang, and B. Li, “Automatic image annotation via compact graph based semi-supervised learning,” *Knowl.-Based Syst.*, vol. 76, pp. 148–165, Mar. 2015.

[22] M. Peikari, S. Salama, S. Nofech-Mozes, and A. L. Martel, "A cluster-then-label semi-supervised learning approach for pathology image classification," *Sci. Rep.*, vol. 8, no. 1, 2018, Art. no. 7193.
[23] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.
[24] N. Idika and A. P. Mathur, "A survey of malware detection techniques," Purdue Univ., West Lafayette, IN, USA, Tech. Rep., 2007, vol. 48.
[25] P. Vadrevu, B. Rahbarinia, R. Perdisci, K. Li, and M. Antonakakis, "Measuring and detecting malware downloads in live network traffic," in *Proc. Eur. Symp. Res. Comput. Secur.* Berlin, Germany: Springer, 2013, pp. 556–573.
[26] L. Page, S. Brin, R. Motwani, and T. Winograd, "The PageRank citation ranking: Bringing order to the Web," Stanford InfoLab, Stanford, CA, USA, Tech. Rep., 1999.
[27] G. H. Golub and C. F. Van Loan, "Matrix computations," *Google Scholar*, pp. 111263–112264, 2012.
[28] U. Dvir, "Security server in the cloud," U.S. Patent 11 462 046, Feb. 15, 2007.
[29] I. Santos, J. Nieves, and P. G. Bringas, "Semi-supervised learning for unknown malware detection," in *Proc. Int. Symp. Distrib. Comput. Artif. Intell.* Berlin, Germany: Springer, 2011, pp. 415–422.
[30] L. Invernizzi, S. Miskovic, R. Torres, C. Kruegel, S. Saha, G. Vigna, S.-J. Lee, and M. Mellia, "Nazca: Detecting malware distribution in large-scale networks," in *Proc. NDSS*, vol. 14, 2014, pp. 23–26.
[31] Z. Gyöngyi, H. Garcia-Molina, and J. Pedersen, "Combating Web spam with trustrank," in *Proc. 13th Int. Conf. Very Large Data Bases VLDB Endowment*, vol. 30, 2004, pp. 576–587.
[32] J. M. Kleinberg, "Authoritative sources in a hyperlinked environment," *J. ACM*, vol. 46, no. 5, pp. 604–632, 1999.
[33] M. Ben Neria, N.-S. Yacovzada, and I. Ben-Gal, "A risk-scoring feedback model for webpages and Web users based on browsing behavior," *ACM Trans. Intell. Syst. Technol.*, vol. 8, no. 4, 2017, Art. no. 53.
[34] J. Huang, Y. Xie, F. Yu, Q. Ke, M. Abadi, E. Gillum, and Z. M. Mao, "SocialWatch: Detection of online service abuse via large-scale social graphs," in *Proc. 8th ACM SIGSAC Symp. Inf., Comput. Commun. Secur.*, 2013, pp. 143–148.
[35] G.-R. Xue, H.-J. Zeng, Z. Chen, W.-Y. Ma, H. Zhang, and C.-J. Lu, "User access pattern enhanced small Web search," Tech. Rep., 2003.
[36] O. Chapelle, J. Weston, and B. Schölkopf, "Cluster kernels for semi-supervised learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2003, pp. 601–608.
[37] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, "Learning with local and global consistency," in *Proc. Adv. Neural Inf. Process. Syst.*, 2004, pp. 321–328.

YITU FU received the M.S. degree in control engineering from the College of Information Science and Engineering, Northeastern University, China, in 2013, where he is currently pursuing the Ph.D. degree in control theory and control engineering. From 2013 to 2015, he was an Engineer with the Equipment Department, BaoWu Steel Group Corporation, Shanghai, China. Since 2018, he has been a Research Scholar with the Department of Industrial Engineering, University of Pittsburgh, Pittsburgh, PA, USA. His current research interests include cloud computing, data mining, machine learning, and artificial intelligence and applications.

JU XU was born in Jiangyin, Jiangsu, China, in 1987. She is currently pursuing the Ph.D. degree in business management with Tongji University, Shanghai, China. From 2018 to 2019, she was a Visiting Scholar with the Department of Organization, Strategy, and International Management, Naveen Jindal School of Management, University of Texas at Dallas. Her research interests include strategic optimization, data mining, cognition, and organizational learning.