Analyzing File-to-File Relation Network in Malware Detection

Lingwei Chen[1], William Hardy[1], Yanfang Ye[1](✉), and Tao Li[2]

[1] Department of Computer Science and Electrical Engineering, West Virginia University, Morgantown, WV 26506, USA
{lgchen,whardy2}@mix.wvu.edu, yanfang.ye@mail.wvu.edu

[2] School of Computer Science, Florida International University, Miami, FL 33199, USA
taoli@cs.fiu.edu

Abstract. Due to its major threats to Internet security, malware detection is of great interest to both the anti-malware industry and researchers. Currently, features beyond file content are starting to be leveraged for malware detection (e.g., file-to-file relations), which provide invaluable insight about the properties of file samples. However, we still have much to understand about the relationships of malware and benign files. In this paper, based on the file-to-file relation network, we design several new and robust graph-based features for malware detection and reveal its relationship characteristics. Based on the designed features and two findings, we first apply Malicious Score Inference Algorithm (MSIA) to select the representative samples from the large unknown file collection for labeling, and then use Belief Propagation (BP) algorithm to detect malware. To the best of our knowledge, this is the first investigation of the relationship characteristics for the file-to-file relation network in malware detection using social network analysis. A comprehensive experimental study on a large collection of file sample relations obtained from the clients of anti-malware software of Comodo Security Solutions Incorporation is performed to compare various malware detection approaches. Promising experimental results demonstrate that the accuracy and efficiency of our proposed methods outperform other alternate data mining based detection techniques.

Keywords: File-to-file relation network · Malware detection · Social network analysis

1 Introduction

Malware (short for ***mal***icious soft***ware***), is software disseminated by an attacker in the hopes of causing harm (e.g., viruses, worms, backdoors, spyware, trojans) [7]. This causes many Internet users emotional and financial troubles, especially when private information is divulged. To put this into perspective, according to a recent CSI survey, the average loss caused by malware attacks is about $345,000 dollars per incident [5]. Thus, the detection of malware is of great interest to both the anti-malware industry and researchers in order to curb the major threats to Internet security it poses.

Currently, most malware detection is done by anti-malware software products (e.g., Symantec, Kaspersky, Comodo), which typically use the signature-based method [8]. A signature is a short sequence of bytes unique to each known malware, which allows newly encountered files to be correctly identified with a small error rate [11]. However, driven by economic benefits, today's malware are created at a rate of thousands per day [28]. Meanwhile, malware disseminators easily evade this method through techniques such as obfuscation, polymorphism, and encryption [19]. In order to remain effective, new, intelligent malware detection techniques need to be investigated. As a result, many research efforts have been conducted on applying data mining techniques for intelligent malware detection [1,13,26,27]. Most existing researchers utilize local features of malware samples, either static or dynamic representations (e.g. binary n-grams [1], system calls [17], or behavior based features [8]), and apply specific classification/clustering methods. Currently, features beyond file content are starting to be leveraged for malware detection [2,10,22,28], such as machine-to-file relations [2] and file-to-file relations [22,28], which provide invaluable insight about the properties of file samples. In our previous work [3,28], we proposed a semi-parametric classification model for combining file content and file relations together for malware detection [28], and then we further adopted Belief Propagation algorithm to detect malware based on the constructed file relation graphs [3].

However, much needs to be done to understand about the relationships of malware and benign files. Based on the file-to-file relationships, current studies [22,28] merely employ the superficial graph properties without delving deeper into the features of the files' mutual relationships as a network. As is investigated, helpful information can be gleaned from the relationships between malware and benign files. Based on the constructed file-to-file relation graphs, it is of interest to know:

- *What graph-based features can be employed for malware detection?*
- *Is the legitimacy of a file affected by indirectly connected files?*
- *Is each malware of equal importance? If not, what are the differences between the important malware and non-important ones?*

In this paper, resting on the constructed file-to-file relation graph, we design several new and robust graph-based features for malware detection and investigate its relationship characteristics. Based on the designed features and two findings, we propose inference learning algorithms composed of Malicious Score Inference Algorithm (MSIA) and Belief Propagation (BP) algorithm to detect malware from unknown file collection. To the best of our knowledge, this is the first work of investigating the relationship characteristics of the file-to-file relation network in malware detection using social network analysis. The major contributions of our work can be summarized as follows:

- *Graph-based feature design for malware detection:* We formalize our malware detection as a problem of analysis and identification for malware's social network and design five graph-based features to represent each file.

- *Provide insight into the relationship characteristics of the file-to-file relation network:* Based on the constructed file-to-file relation graph, we further analyze its relationship characteristics and have two findings: ***Finding 1:*** A file can greatly inherit the indirect influences from other files. ***Finding 2:*** (1) The importance of each file is different; (2) The neighbors of the important malware are associated through it, while the neighbors of the non-important malicious file are inclined to be a clique. We also use graph metrics to quantitatively validate these findings (See Sect. 3.3 for details).
- *Build effective graph inference algorithms for malware detection:* Based on the designed features and two findings, we first apply Malicious Score Inference Algorithm (MSIA) to select the representative samples from the large unknown file collection for labeling, and then use Belief Propagation (BP) algorithm to detect malware.
- *Comprehensive experimental study on a real dataset from an anti-malware industry company:* Resting on $7,093$ clients, we obtain the file relations between $9,893$ malware samples, $19,402$ benign files, and $31,429$ unknown files from Comodo Cloud Security Center. We then build a practical solution for malware detection based on our proposed algorithms and provide a comprehensive experimental study.

The rest of the paper is organized as follows. Section 2 discusses the related work. Section 3 describes the designing of graph-based features for malware detection and reveals relationship characteristics of the file-to-file relation network. Section 4 introduces graph inference algorithms for malware detection. In Sect. 5, we systematically evaluate the effectiveness and efficiency of our proposed method in comparison with other alternate methods for malware detection. Finally, Sect. 6 concludes.

2 Related Work

In the area of malware detection, limited researches have been done using features beyond file content [2,3,22,28], such as machine-to-file relations [2] and file-to-file relations [22,28], which provide invaluable insight about the properties of file samples. However, these researches still lack insights into the characteristics of file-to-file relation network.

An Online Social Network (OSN) is a convenient communication platform for Internet users which creates an indispensable environment. However, some users exploit this platform for nefarious goals, e.g. propagating advertisements, reposting duplicate microblogs, or following a large number of users to gain more themselves [12,14,23]; these users are considered spammers and parallel malicious files in our domain. Many research efforts have been devoted to spammer detection [9,12,15,20,25] for OSNs, which successfully set examples in designing robust graph-based features for spammer detection [24], and understanding the characteristics of criminal accounts' social relationships [25], etc.

Yang et al. [24] proposed several robust graph-based features, such as local clustering coefficient, betweenness centrality, and bi-directional link ratio, to detect Twitter spammers. The reasoning for these features was that it was difficult for spammers to change their positions in the graph even when they change their behaviors. Ting et al. [21] represented several frequently used technologies in social networks analysis where degree centrality and closeness centrality were used to specify each node's importance. Chen et al. [4] proposed coefficients of reposting and commenting to detect the user influence when a new post was published. Steep rise in reposting or commenting implied the account was legitimate, otherwise spammer.

Using these features, a number of spammer detection methods were designed. In [9,12,24,30], classical data mining based methods (e.g., Support Vector Machine, Decision Tree, and Naïve Bayes) were applied for spammer detection. Moh et al. [15] used trust propagation to infer whether a user is a spammer or a legitimate user. Tang et al. [20] proposed trust index utilizing the idea of page rank for criminal account detection. Yang et al. [25] proposed a malicious relevance score propagation algorithm to extract criminal supporters. Hu et al. [9] presented a unified model to effectively integrate both social network information and content information for spammer detection.

Different from previous work on malware detection [22,28], based on the file-to-file relation network, we would like to know more about its relationship characteristics. The researches on spammer detection for OSNs have inspired us to obtain the solutions and we attempt to transfer some of the researches done on spammer detection for malware detection.

3 Investigation of File-to-File Relation Network

In this section, we empirically analyze the malware's characteristics based on the constructed file-to-file relation network using the data collected from Comodo Cloud Security Center by: (1) visualizing its relationship graph, (2) designing its graph-based features, and (3) revealing its relationship characteristics.

3.1 Visualization of File Relation Graph

To achieve our research goals, we analyze the dataset obtained from Comodo Cloud Security Center, which contains the relationships between $60,724$ files (9,893 malware, 19,402 benign files and 31,429 unknown files) on $7,093$ clients. Based on the collected data, we construct a file-to-file relation graph to describe the relations among file samples. Generally, two files are related if they are shared by many clients (or equivalently, file lists). The social graph is defined as $G = (V, E)$, where V is the set of file samples and E denotes the relations between file samples. Given two file samples v_i and v_j, let C_i be the set of clients containing v_i and C_j be the set of clients containing v_j. $|.|$ represents the size of a set. The connectivity between v_i and v_j is computed as

$$con(v_i, v_j) = \frac{|C_i \bigcap C_j|}{|C_i \bigcup C_j|}. \tag{1}$$

If the connectivity between a pair of file samples is greater than the specified threshold (e.g., 0), then there is an edge between them. Each file is in a state $S \in \{s_m, s_b, s_u\}$ (s_m: malicious, s_b: benign, s_u: unknown). Assume that v_i is in state s_i and v_j is with state s_j, the weight of the edge between v_i and v_j which infers the probability that node i and node j can be connected together is defined as

$$w(v_i, v_j) = \frac{\left|E_{s_i,s_j}\right|}{|E|}, \tag{2}$$

where $\left|E_{s_i,s_j}\right|$ is the number of the edges between all the files with states s_i and s_j, and $|E|$ is the number of all the edges. The weight of node v_i which denotes its popularity can be defined as

$$w(v_i) = \frac{|C_i|}{|C|}, \tag{3}$$

where C is the set of all the clients.

For the file relations collected from 7,093 clients, we construct the graph consisting of 60,724 nodes and 3,471,288 edges. Figure 1(a) shows a part of the constructed graph, while Fig. 1(b) and (c) give examples of a malware relation graph and a benign file relation graph with one-hop information respectively.

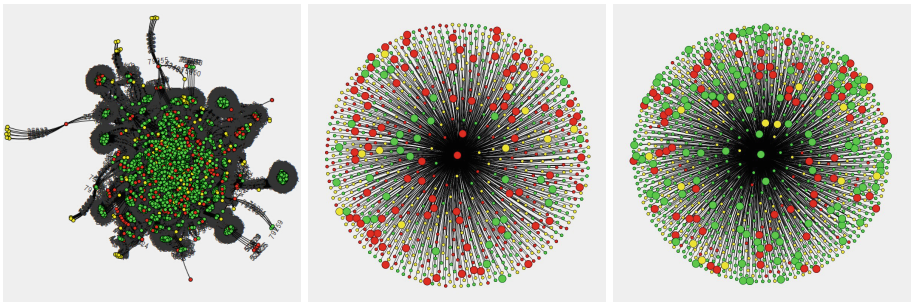

(a) File-to-File Relation Graph (b) Malware Relation Graph (c) Benign File Relation Graph

Fig. 1. Visualization of file-to-file relation graphs. (Red nodes denote malware, green nodes represent benign file, and yellow nodes are unknown file.) (Color figure online)

3.2 Designing Graph-Based Features for Malware Detection

To counter malware's evasion tactics, after the construction of the file-to-file relation graph, we further investigate several robust graph-based features for malware detection. Ideal features are either difficult or costly to evade, even when malware is obfuscated. In this section, on the basis of special characteristics of the file relationships between malware and benign files, we design five robust and representative graph-based features for malware detection, which are described in details in the followings.

Vertex Degree. The degree of a vertex in a graph is the number of edges incident to the vertex, which can specifically represent the association between the vertex and its neighbors [6]. In the file relation graph, we use the degree of malware (DoM) and degree of benign files (DoB) to capture the association between the file and its neighbors. These two metrics can be calculated as

$$DoM(v) = |\delta_m^v|,\ DoB(v) = |\delta_b^v|, \tag{4}$$

where $|\delta_m^v|$ is the total number of vertex v's malicious neighbors, and $|\delta_b^v|$ is the total number of vertex v's benign neighbors. As the moral says that "man is known by the company he keeps", it's easy to understand that malware is more likely to have a larger DoM than DoB, and vice versa. To further support this point, we calculate the degree for each file in the collected dataset described above: 53.75 % of malware have larger DoM than DoB; while only 3.10 % of benign files have larger DoM than DoB.

Influence Coefficient. For spammer detection, in [4], the authors used reposting and commenting coefficients to indicate the ability that a user affects others to repost or comment. In malware detection, we define the influence coefficient of malware and benign files by Eqs. 5 and 6.

$$IoM(v) = \frac{\sum_{i=1}^{N} \log(Malware\_Count(v_i) + 1)}{N}, \tag{5}$$

$$IoB(v) = \frac{\sum_{i=1}^{N} \log(Benign\_Count(v_i) + 1)}{N}, \tag{6}$$

where N denotes the number of vertex v's neighbors and v_i denotes the i^{th} neighbor of v. $Malware\_Count(v_i)$ and $Benign\_Count(v_i)$ represent the number of the malware and benign files directly connected to v_i respectively. A file can directly or indirectly inherit the goodness or malice from other files. Compared with vertex degree, which considers the information one-hop away from the node, the feature of influence coefficient takes the indirect influence from other files into consideration.

Local Clustering Coefficient. The local clustering coefficient of a vertex in a graph specifies how close vertices in its neighborhood are to being a clique [24]. For each vertex in the constructed file relation graph, its local clustering coefficient can be calculated as [24]

$$LCC(v) = \frac{2|e^v|}{k_v(k_v - 1)}, \tag{7}$$

where $|e^v|$ is the total number of edges built by all v's neighbors, and k_v is the degree of the vertex v. For benign files, different users may install different sets of applications according to their occupations, ages, etc. And these applications are unnecessary to have associations with each other. However, for malware, just specified groups of users would be infected by malware. When infected, not only one malicious software would appear in the client, but also its related files would be released or downloaded. For example, variants of trojans will always come together with trojan-downloader and co-exist in the clients. Therefore, malware will have a larger local clustering coefficient than benign files. To quantitatively validate this, we calculate the local clustering coefficient for each file in the collected dataset described above: the average LLC for malware is 0.9387, while the average LLC for benign files is 0.7573.

Degree Centrality. Degree centrality of a vertex is determined by the number of vertices adjacent to it. The larger the degree, the more important the vertex is [18]. In malware detection, degree centrality can be used to quantify the importance of a file, which can be computed as [18]

$$DC(v) = \frac{\delta(v)}{n-1}, \tag{8}$$

where $\delta(v)$ is the degree of the vertex v, and n is the number of vertices in the graph.

Closeness Centrality. Closeness centrality measures the importance of vertices by quantifying their centrality. Central vertices tend to reach the whole graph more quickly than non-central vertices [18]. Closeness centrality factors in how close a vertex is to other vertices, which is computed as [18]

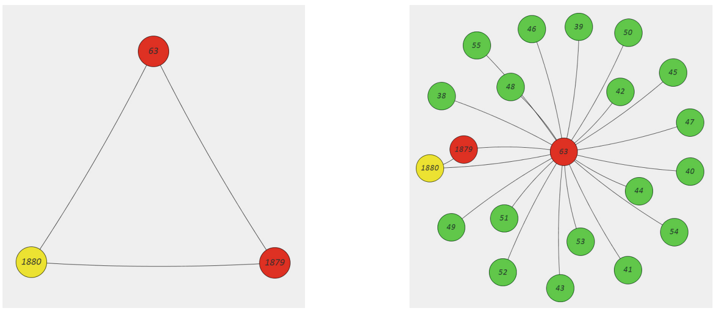
$$CC(v) = \frac{1}{n-1} \sum_{u \neq v}^{n} g(u,v), \tag{9}$$

where $g(u,v)$ is the distance between the vertex u and vertex v, and n is the number of vertices in the graph. Malware attackers always use a shotgun approach to find victims and allure them to download variants of malicious files (e.g., trojans, adware). These files in the victim clients are always connected through the downloaders. Thus, the closeness centrality of those downloaders will be high.

3.3 Revealing Relationship Characteristics

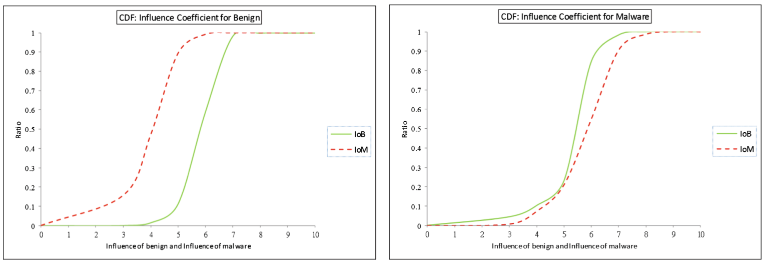
After visualizing the constructed file-to-file relation network and designing the graph-based features, we further analyze its relationship characteristics, and give the following observations.

Finding 1: A file can greatly inherit the indirect influences from other files in the file-to-file relation network. Again, as the moral says "man is known by the company he keeps", in malware detection, a file's goodness or malice can be judged by the other files that always co-exist with it in the clients.



(a) The direct influences from its neighbors (b) The indirect influences from other files

Fig. 2. The indirect influences superior than direct influences for file 1880 (yellow node) (Color figure online)



(a) Benign files (b) Malware

Fig. 3. The comparison of benign files and malware in IoB and IoM measures

However, sometimes, a file can not only be directly influenced by its neighbors, but also greatly inherit the influences from other files (e.g., its neighbors' neighbors). Figure 2 shows an example that the indirect influences is superior than the direct influences for file 1880 (marked in yellow node). To quantitatively validate this finding, we use the features of influence of benign files (IoB) and influence of malware (IoM) designed in the above section for measure. For file 1880, its IoB is 1.6290, while its IoM is just 0.6931, which means this file is more likely to be influenced by benign files, even though all the files it directly connects with are malware.

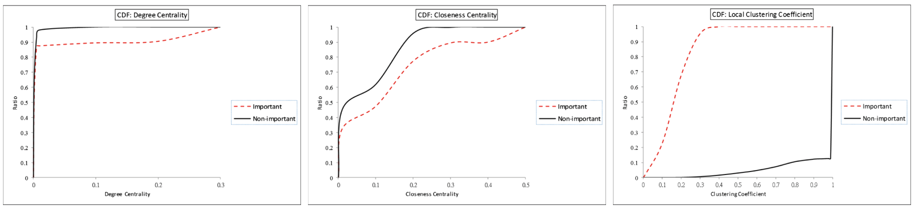
To further illustrate, based on the collected dataset described in Sect. 3.1, we measure the indirect influences from other files for each node. Figure 3 displays the Cumulative Distribution Function (CDF) of IoB and IoM for both malware and benign files, which shows that both benign files and malware can greatly inherit the goodness and malice indirectly from other files.

Possible Factor: To disseminate the malicious files, it is not uncommon for malware to be packaged into a software product (especially when it is free and open source) by the attackers. This would cause such kind of benign software to be closely related to malicious files, however, their neighbors of neighbors would not necessarily be. On the other hand, variants of online game trojans may have indirect associations through the same kind of online game applications, since they target on stealing specific kind of online game accounts' information, but they are unnecessary to co-exist in the same clients.

From the observation above, we can see that a file's goodness or malice not only depends on its neighbors, but also greatly inherit the indirect influences from other files (e.g., its neighbors' neighbors). Furthermore, we are also interested to know: *(1) Is each malware of equal importance? (2) If not, what are the differences between the important malware and non-important ones?*

Finding 2: In the file-to-file relation network, (1) the importance of each file is different; (2) the neighbors of the important malware are associated through it, while the neighbors of the non-important malicious file are inclined to be a clique. To initially evaluate the importance of each node, we use degree centrality for measure. Based on the collected dataset described in Sect. 3.1, we calculate the degree centrality of each file: about 2 % of the malware have the degree centrality over 0.01, which are 10 – 1000 times larger than the remaining 98 % ones. From this analysis, we can see that the importance of each malware is different: the larger the degree, the more important the vertex is [18]. Note that there is another interesting observation that those malware with larger degrees also have higher node weight values in the graph, which means the "important" malware are always with higher popularity. We mark those 2 % malicious files with higher degree centrality as "important" malware, compared with the remaining 98 % ones. Figure 4(a) displays the CDF of degree centrality for the important malware and non-important ones.

To further analyze the different characteristics of the important and non-important malware, we take insights to their graph structures. Figure 5(a) illustrates an example of the relationship between an important malware A and its neighbors, while Fig. 5(b) shows the relations between its neighbors. From Fig. 5(a) and (c), we can see that both important and non-important malicious nodes with one-hop information have the star-structures, but the degree centralities of them are different. From Fig. 5(b) and (d), we can see that, the neighbors of the important malware are associated through it (the closeness centrality of the important malware A is 0.25), while the neighbors of the non-important



(a) Degree Centrality (b) Closeness Centrality (c) Clustering Coefficient

Fig. 4. The comparisons of "important" malware and "non-important" ones.

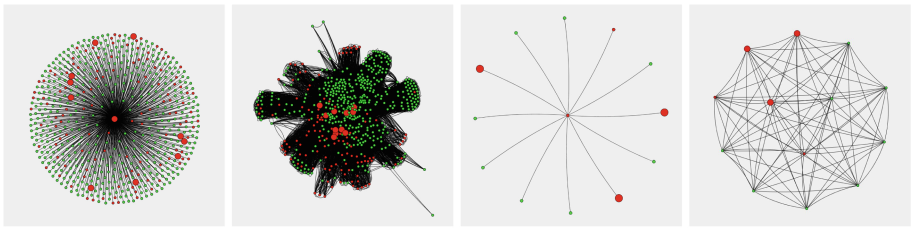


Fig. 5. Graph structure comparisons of "important" and "non-important" malware. (a) An important malware A and its neighbors; (b) File relations between A's neighbors; (c) A non-important malware B and its neighbors; (d) File relations between B's neighbors.

malicious file are inclined to be a clique (the local clustering coefficient LLC of it is equal to 1). Figure 4(b) and (c) display the CDF of local clustering coefficient and closeness centrality for the important malware and non-important ones respectively, which also validate the *Finding 2*.

Possible Factor: The importance of each malware is different, since the impacts different malicious files play are different. For example, a popular trojan or adware downloader will infect more clients, compared with the specific kind of trojan or adware variants. The files co-exist with the popular downloader in different clients are unnecessary to have a close relationship among them, but are associated through the downloader; while the files co-exist with the variants of same trojan or adware are prone to be a clique, since they tend to be the same or similar kind of applications those trojans or adware target on.

4 Inference Learning Algorithms for Malware Detection

Via empirical analysis for the file-to-file relation network, each node v_i (i.e., a file sample) in the constructed graph can be represented by its relations with other nodes and its graph-based features designed in Sect. 3.2, denoted as $Fv_i = \langle Rv_i, Gv_i \rangle$. Rv_i can be defined as

$$Rv_i = \langle v_{1i}, v_{2i}, ..., v_{ni} \rangle, \tag{10}$$

where $v_{ji} = \{0, 1\}$ (i.e., if $(v_j, v_i) \in E$, $v_{ji} = 1$; otherwise, $v_{ji} = 0$). Gv_i can be defined as

$$Gv_i = \langle DoM(v_i), DoB(v_i), IoM(v_i), IoB(v_i), LCC(v_i), DC(v_i), CC(v_i) \rangle. \tag{11}$$

4.1 Extracting Representative Samples from Unknown File Collection

Based on the *Finding 2* that the importance of each file is different and the neighbors of the important malware are associated through them, therefore, selecting those representative malware from the large unknown file collection for labeling is very important to further improve the detection accuracy. In spammer detection, Yang et al. [25] proposed a Malicious Relevance Score Propagation Algorithm (Mr. SPA) to extract criminal supporters. In this section, we propose a Malicious Score Inference Algorithm (MSIA), which adapts and improves Mr. SPA [25] to assign a malicious score for each file to quantify its importance.

Given a constructed file relation graph $G = (V, E)$, let n be the number of nodes (files) in the graph, and $I(v_i, v_j)$ be the indicator to denote whether $(v_i, v_j) \in E$ (i.e., if $(v_i, v_j) \in E$, $I(v_i, v_j) = 1$; otherwise, $I(v_i, v_j) = 0$). At each step, for each node v_i, its malicious score $M(v_i)$ can be calculated as [25]

$$M(v_i) = \alpha \cdot \sum_{j=1}^{n} I(v_i, v_j) W(v_i, v_j) M(v_j), \tag{12}$$

where α is an adjustable factor, and $W(v_i, v_j)$ is the weight between v_i and v_j which reflects the coordination between each pair of nodes. For each node v_i, we calculate the similarity between itself and each of its neighbors v_j based on their presented features described above, denoted as $sim(Fv_i, Fv_j)$. Then, the weight $W(v_i, v_j)$ between node v_i and v_j is computed as

$$W(v_i, v_j) = \frac{sim(Fv_i, Fv_j)}{\sum_{e_{v_k v_j} \in E} sim(Fv_k, Fv_j)}. \tag{13}$$

In our application, we initialize $M^0(v_i) = \{0, 1\}$ (i.e., if v_i is malicious, $M^0(v_i) = 1$; otherwise, $M^0(v_i) = 0$).

With the consideration of the historical score record for each node, at each step $t (t \geq 1)$, an initial score bias $(1 - \alpha) \cdot M_i^0$ is added to its malicious score. Thus the malicious score vector $\overrightarrow{\boldsymbol{M}^t}$ for all nodes at step $t (t \geq 1)$ can be computed as [25]

$$\overrightarrow{\boldsymbol{M}^t} = \alpha \cdot \overrightarrow{\boldsymbol{I} \cdot \boldsymbol{M}^{t-1}} + (1 - \alpha) \cdot \overrightarrow{\boldsymbol{M}^0}. \tag{14}$$

The algorithm MSIA stops when the updates of malicious score vector converge or a maximum number of the iterations has finished. The higher the malicious score it has, the more important the file is.

4.2 A Belief Propagation Algorithm for Malware Detection

Based on *Finding 1*, which states a file's goodness or malice not only depends on its neighbors, but also indirectly on other files (e.g., its neighbors' neighbors), we apply Belief Propagation (BP) algorithm for malware detection, since BP algorithm can propagate the indirect influences from other files for each node. BP algorithm is a promising method for solving inference problems over graphs and it has also been successfully used in many domains (e.g., computer vision, coding theory) [29]. Nodes of the graph perform as a local summation operation by iterations using the prior knowledge from their neighbors and then pass the information to all the neighbors in the form of messages [16]. In our previous work [3], we proposed a variant of Belief Propagation (BP) algorithm for malware detection. The message update equation is defined as [3]

$$m_{i->j}(v_j) = \frac{1}{\beta} \sum_{v_i \in S} f_{i->j}(v_i, v_j) g_i(v_i) \frac{\sum_{k \in N(i)/j} m_{k->i}(v_i)}{p}, \tag{15}$$

where $m_{i->j}(v_j)$ is the probability node i believes that the neighbor node j being a benign file; $f_{i->j}(v_i, v_j)$ is the probability that node i and node j can be connected together, which is equal to the weight of edge described in Eq. 2; $g_i(v_i)$ is the prior probability node i being a benign file [3]; p equals to the number of the neighbors of node i (excluded node j) and β is a normalizing constant. The belief read-out equation is designed as follow [3]

$$b_i(v_i) = \frac{1}{\gamma} g_i(v_i) \frac{\sum_{k \in N(i)} m_{k->i}(v_i)}{p}, \tag{16}$$

where γ is an adjustable constant.

5 Experimental Results and Analysis

In this section, we conduct three sets of experiments based on the collected dataset obtained from Comodo Cloud Security Center described in Sect. 3.1: (1) In the first set of experiments, we use MSIA to evaluate the effectiveness of the designed features; (2) In the second set of experiments, we further evaluate our proposed inference learning algorithms in malware detection; (3) In the last set of experiments, we compare our proposed algorithms with other classification methods (i.e., SVM, Decision Tree, and Naïve Bayes).

We evaluate the malware detection performance of different methods using the measures of True Positive (TP), True Negative (TN), False Positive (FP), False Negative (FN), and Accuracy (ACY) [3]. All the experiments are conducted under the environment of 64 Bit Windows 7 operating system with 4th Generation Intel Core i7 Processor (Quad Core, 8 MB Cache, up to 4.0 GHz w/Turbo Boost) plus 16 G of RAM using Apache Pig, MySQL and C++.

5.1 Evaluation of the Designed Features

For each sample in the constructed file-to-file relation network, we extract its relations with other samples described in Sect. 3.1 and its graph-based features designed in Sect. 3.2 for representation. In this section, we conduct the experiments using MSIA to evaluate the effectiveness of the designed features. The collected dataset from Comodo Cloud Security Center contains 60,724 files: 9,893 are malware, 19,402 are benign files, and 31,429 are unknown (with the analysis by anti-malware experts of Comodo Security Lab, 470 of them are labeled as malware and 1,273 of them are benign files). Those 9,893 malware and 19,402

Table 1. Evaluation of the designed graph-based features

Training	TP	FP	TN	FN	ACY
MSIA(RF)	7,392	1,301	18,101	2,501	0.8702
MSIA(GF)	6,960	2,410	16,992	2,933	0.8176
MSIA($RF + GF$)	7,890	1,371	18,031	2,003	0.8848
Testing	TP	FP	TN	FN	ACY
MSIA(RF)	293	78	1,195	177	0.8537
MSIA(GF)	179	122	1,151	291	0.7631
MSIA($RF + GF$)	315	78	1,195	155	0.8663

Remark 1. RF denotes the file relation features of the samples, while GF denotes the graph-based features of the samples.

benign files are used for training, while 470 malware and 1,273 benign files from the unknown file collection which are labeled by anti-malware experts are used for testing. The results in Table 1 demonstrate the effectiveness of the designed features in malware detection.

5.2 Evaluation of the Proposed Inference Learning Algorithms in Malware Detection

In this section, we further evaluate our proposed inference learning algorithms in malware detection: (1) Based on the training and testing sets described in Sect. 5.1, we compare the performance of MSIA and BP in malware detection; (2) To further improve the detection accuracy, we first apply MSIA for representative samples selection (193 samples are selected from the unknown file collection for labeling), and then use BP for detection. The results in Table 2 show that our proposed algorithms composed of MSIA and BP (MSIA+BP) can greatly improve the accuracy in malware detection, compared with using MSIA and BP respectively, or BP after randomly selecting 193 samples from the unknown file collection for labeling (Random+BP).

5.3 Comparisons of the Proposed Algorithms with Other Classification Methods

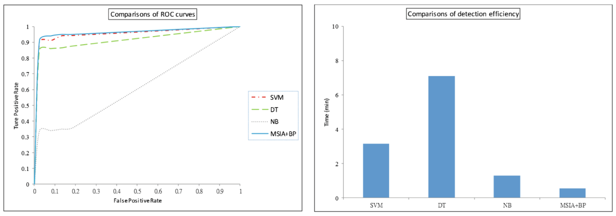
In this section, we further compare our proposed algorithms with other classification methods (Support Vector Machine (SVM), Decision Tree (DT), and Naïve Bayes (NB)) based on the same testing dataset described in Sect. 5.1. The results in Table 3 and the ROC curves based on the testing set in Fig. 6(a) demonstrate that our proposed algorithm composed of MSIA and BP (MSIA+BP) is superior to SVM, DT, and NB in malware detection. Figure 6(b) shows that the detection efficiency of our proposed algorithms outperform other classification methods.

Table 2. Evaluation of the proposed inference learning algorithms in malware detection

Training	TP	FP	TN	FN	ACY
MSIA	7,890	1,371	18,031	2,003	0.8848
BP	9,881	866	18,536	12	0.9700
Random+BP	10,059	870	18,545	14	0.9701
MSIA+BP	10,060	851	18,564	13	0.9707
Testing	TP	FP	TN	FN	ACY
MSIA	315	78	1,195	155	0.8663
BP	411	119	1,154	59	0.8979
Random+BP	462	204	1,069	8	0.8784
MSIA+BP	437	100	1,173	33	0.9236

Table 3. Comparisons of different detection methods

Training	TP	FP	TN	FN	ACY
SVM	8,661	797	18,618	1,412	0.9251
DT	8,308	1,761	17,654	1,765	0.8804
NB	5,288	502	18,913	4,785	0.8207
MSIA+BP	10,060	851	18,564	13	0.9707
Testing	TP	FP	TN	FN	ACY
SVM	452	172	1,101	18	0.8910
DT	412	241	1,032	58	0.8285
NB	159	31	1,242	311	0.8037
MSIA+BP	437	100	1,173	33	0.9236



(a) ROC curves (b) Detection Efficiency

Fig. 6. Comparisons of ROC curves and detection efficiency of different methods

6 Conclusion and Future Work

In this paper, resting on the constructed file-to-file relation graphs, we design several new and robust graph-based features for malware detection and reveal the characteristics of file relation network. Based on the designed features and two findings, we propose effective inference learning algorithms for malware detection. To the best of our knowledge, this is the first investigation of the relationship characteristics for the file-to-file relation network in malware detection using social network analysis. Due to the difficulty in thorougly obtaining the social interactions and motivations of malware, we recognize that the validations on some proposed explanations are not entirely rigorous. However, we believe that our novel analysis of those phenomena still yields great value and unveils a new avenue for better understanding malware's file relation ecosystem. In our future work, we will further investigate the distinctions between malware and benign files in the relationship graph, as well as design a full detection solution by combining file relations with other detection features to further improve the detection accuracy.

Acknowledgments. The authors would also like to thank the anti-malware experts of Comodo Security Lab for the data collection, as well as the helpful discussions and supports.

References

1. Bailey, M., Oberheide, J., Andersen, J., Mao, Z.M., Jahanian, F., Nazario, J.: Automated classification and analysis of internet malware. In: Kruegel, C., Lippmann, R., Clark, A. (eds.) RAID 2007. LNCS, vol. 4637, pp. 178–197. Springer, Heidelberg (2007)
2. Chau, D., Nachenberg, C., Wilhelm, J., Wright, A., Faloutsos, C.: Polonium: terascale graph mining for malware detection. In: SIAM International Conference on Data Mining (SDM), pp. 131–142 (2011)
3. Chen, L., Li, T., Abdulhayoglu, M., Ye, Y.: Intelligent malware detection based on file relation graphs. In: 9th IEEE International Conference on Sematic Computing, pp. 85–92 (2015)
4. Chen, K., Zhu, P., Xiong, Y.: Mining spam accounts with user influence. In: International Conference on ISCC-C, pp. 167–173 (2013)
5. Computer Security Institute: 12th annual edition of the CSI computer crime and security survey. Technical report, Computer Security Institute (2007)
6. Diestel, R.: Graph Theory, vol. 173, 4th edn. Springer, Heidelberg (2010)
7. Egele, M., Scholte, T., Kirda, E., Kruegel, C.: A survey on automated dynamic malware analysis techniques and tools. ACM CSUR **44**(2), 6:1–6:42 (2008)
8. Filiol, E., Jacob, G., Liard, M.L.: Evaluation methodology and theoretical model for antiviral behavioural detection strategies. J. Comput. Virol **3**(1), 27–37 (2007)
9. Hu, X., Tang, J., Zhang, Y., Liu, H.: Social spammer detection in microblogging. In: Proceedings of the 23rd IJCAI, pp. 2633–2639 (2013)
10. Karampatziakis, N., Stokes, J.W., Thomas, A., Marinescu, M.: Using file relationships in malware classification. In: Flegel, U., Markatos, E., Robertson, W. (eds.) DIMVA 2012. LNCS, vol. 7591, pp. 1–20. Springer, Heidelberg (2013)
11. Kephart, J., Arnold, W.: Automatic extraction of computer virus signatures. In: Proceedings of 4th Virus Bulletin International Conference, pp. 178–184 (1994)
12. Lin, C., Zhou, Y., Chen, K., He, J., Yang, X., Song, L.: Analysis and identification of spamming behaviors in Sina Weibo microblog. In: SNAKDD 2013 (2013)

13. Masud, M.M., Al-Khateeb, T.M., Hamlen, K.W., Gao, J., Khan, L., Han, J., Thuraisingham, B.: Cloud-based malware detection for evolving data streams. ACM TMIS **2**(3), 16:1–16:27 (2008)
14. Mislove, A., Marcon, M., Gummadi, K.P., Druschel, P., Bhattacharjee, B.: Measurement and analysis of online social networks. In: Proceedings of the 7th ACM SIGCOMM, pp. 29–42 (2007)
15. Moh, T.-S., Murmann, A.J.: Can you judge a man by his friends? Enhancing spammer detection on the twitter microblogging platform using friends and followers. In: Prasad, S.K., Vin, H.M., Sahni, S., Jaiswal, M.P., Thipakorn, B. (eds.) ICISTM 2010. CCIS, vol. 54, pp. 210–220. Springer, Heidelberg (2010)
16. Noorshams, N., Wainwright, M.J.: Belief propagation for continuous state spaces: stochastic message-passing with quantitative guarantees. J. Mach. Learn. Res. **14**(1), 2799–2835 (2013)
17. Park, Y., Zhang, Q., Reeves, D., Mulukutla, V.: AntiBot: clustering common semantic patterns for bot detection. In: IEEE 34th Annual Computer Software and Applications Conference, pp. 262–272 (2010)
18. Scott, J.: Social Networks Analysis: A Hand Book, 2nd edn. SAGE Publications Ltd, Thousand Oaks (2000)
19. Sung, A., Xu, J., Chavez, P., Mukkamala, S.: Static analyzer of vicious executables (save). In: Proceedings of the 20th ACSAC, pp. 326–334 (2004)
20. Tang, R., Lu, L., Zhuang, Y., Fong, S.: Not every friend on a social network can be trusted: an online trust indexing algorithm. In: IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology (WI-IAT), pp. 280–285 (2012)
21. Ting, I.H., Wang, S.L.: Content matters: a study of hate groups detection based on social networks analysis and web mining. In: IEEE/ACM ASONAM, pp. 1196–1201 (2013)
22. Tamersoy, A., Roundy, K.A., Chau, D.: Guilt by association: large scale malware detection by mining file-relation graphs. In: ACM SIGKDD (2014)
23. Weng, J., Lim, E.P., Jiang, J., He, Q.: Twitterrank: finding topic-sensitive influential twitterers. In: Proceedings of the Third ACM WSDM, pp. 261–270 (2010)
24. Yang, C., Harkreader, R.C., Gu, G.: Die free or live hard? Empirical evaluation and new design for fighting evolving twitter spammers. In: Proceedings of the 14th International Conference on Recent Advances in Intrusion Detection, pp. 318–337 (2011)
25. Yang, C., Harkreader, R., Zhang, J., Shin, S., Gu, G.: Analyzing spammer's social networks for fun and profit: a case study of cyber criminal ecosystem on twitter. In: Proceedings of the 21st International Conference on World Wide Web (WWW 2012), pp. 71–80 (2012)
26. Ye, Y., Wang, D., Li, T., Ye, D., Jiang, Q.: An intelligent PE-malware detection system based on association mining. J. Comput. Virol. **4**, 323–334 (2008)
27. Ye, Y., Wang, D., Li, T., Ye, D.: IMDS: Intelligent malware detection system. In: Proceedings of the 13th ACM SIGKDD, pp. 1043–1047 (2007)
28. Ye, Y., Li, T., Zhu, S., Zhuang, W., Tas, E., Gupta, U., Abdulhayoglu, M.: Combining file content and file relations for cloud based malware detection. In: Proceedings of the 17th ACM SIGKDD, pp. 222–230 (2011)
29. Yedidia, J. S., Freeman, W.T., Weiss, Y.: Understanding belief propagation and its generalizations. Mltsubishl Electric Research Laboratories (2001)
30. Zhang, C., Niu, K., He, Z.: Dynamic detection of spammers in Weibo. In: 4th IEEE IC-NIDC, pp. 112–116 (2014)