Deep Analysis and Utilization of Malware's Social Relation Network for Its Detection

Shifu Hou[1], Lingwei Chen[1], Yanfang Ye[1(✉)], and Lifei Chen[2]

[1] Department of Computer Science and Electrical Engineering, West Virginia University, Morgantown, WV 26506, USA
{shhou,lgchen}@mix.wvu.edu, yanfang.ye@mail.wvu.edu
[2] School of Mathematics and Computer Science, Fujian Normal University, Fuzhou 350117, Fujian, China
clfei@fjnu.edu.cn

Abstract. To combat with the evolving malware attacks, many research efforts have been conducted on developing intelligent malware detection systems. In most of the existing systems, resting on the analysis of file contents extracted from the file samples (e.g., binary n-grams, system calls), data mining techniques such as classification and clustering have been used for malware detection. However, ignoring the social relations among these file samples (i.e., utilizing file contents only) is a significant limitation of these malware detection methods. In this paper, (1) instead of using file contents extracted from the collected samples, we conduct deep analysis of the social relation network among file samples and study how it can be used for malware detection; (2) resting on the constructed file relation graph, we perform large scale inference by propagating information from the labeled samples (either benign or malicious) to detect newly unknown malware. A comprehensive experimental study on a large collection of file sample relations obtained from Comodo Cloud Security Center is performed to compare various malware detection approaches. Promising experimental results demonstrate that the accuracy and efficiency of our proposed method outperform other alternate data mining based detection techniques.

Keywords: Malware detection · Social relation network · Graph inference

1 Introduction

Nowadays, as computers and Internet become increasingly ubiquitous, especially the rapid development of e-commerce, computer security becomes more and more important. Malware (short for ***mal***icious soft***ware***) is software that deliberately fulfills the harmful intent of an attacker [3], such as viruses, trojans, worms and botnets. It has been used as the major weapon by the cyber-criminals to launch a wide range of security attacks which present serious damages and significant financial loss to Internet users [10]: the average infected computers per day was

© Springer International Publishing AG 2017
S. Song et al. (Eds.): APWeb-WAIM 2017 Workshops, LNCS 10612, pp. 31–42, 2017.
https://doi.org/10.1007/978-3-319-69781-9_4

between 2–5 millions [14] and the average loss caused by malware attacks was around $345,000 dollars per incident [11]. To protect legitimate users from the attacks, the most significant line of defense against malware is anti-malware software products, such as Comodo, Kaspersky and Symantec Anti-Virus. Typically, these widely used malware detection software tools use the signature-based method [4] to recognize threats. However, driven by considerable economic benefits, malware attackers have invented automated malware development toolkits (such as Zeus [23]) to create and mutate thousands of malicious codes per day which can bypass the traditional signature-based detection. In order to effectively and automatically detect these large new generated malware samples, intelligent malware detection systems have been developed by applying data mining techniques [2,13,15,21,26,28,29]. Such techniques have successes in classifying or clustering particular sets of malware samples.

Although utilizing file contents only, either static or dynamic extractions, and simply treating the files as independent samples may allow many off-the-shelf classification or clustering tools to be directly adapted for malware detection, ignoring the social relations among file samples can be a significant limitation of current malware detection methods, since the usual ***i.i.d*** (independent and identical distributed) assumption may not hold for malware samples. Actually, the social relations among the file samples (e.g., whether the files co-exist in the users' computers, whether the files are created at the same time, etc.) may imply the inter-dependence among them and can provide invaluable information about their properties. For example, if a file is always associated with many trojans in users' computers, then most likely, it is a malicious Trojan-Downloader file. In this paper, instead of using file contents extracted from the collected samples, we conduct deep analysis of the social relation network among file samples and study how it can be used for malware detection. Based on the constructed file relation graph, large scale inference by propagating information from the labeled samples (either benign or malicious) is performed to detect newly unknown malware. A comprehensive experimental study on a large collection of file sample relations obtained from Comodo Cloud Security Center is performed to compare various malware detection approaches. Promising experimental results demonstrate that the accuracy and efficiency of our proposed method outperform other alternate data mining based detection techniques. The contributions of this paper can be summarized as follows:

- *Deep Analysis of Malware's Social Relation Network:* Different from file content based detection, we analyze and utilize the social relations among file samples (i.e., co-existences of the files) collected from the user clients to construct file relation graph for malware detection. The newly unknown malware can be detected by its association with the known files (benign or malicious).
- *An Improved Graph Inference Algorithm for Unknown File Labeling:* Belief Propagation (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) [30]. However, in our application, the algorithm should be greatly adapted, which is not a trivial process: we fine tune

various components used in the algorithm and carefully design the message update and belief read-out functions for malware detection.
– *A Comprehensive Experimental Study based on Real Data Collection from Industry:* Based on the real sample set and the co-existence relationship among the files obtained from Comodo Cloud Security Center, we construct the file relation graph and provide a comprehensive experimental study to evaluate our proposed method.

The rest of the paper is organized as follows. Section 2 discusses the related work. Section 3 presents file relation graph construction method and provides a deep analysis of malware's social relation network. Section 4 introduces our proposed graph inference algorithm for malware detection based on the constructed file relation graph. In Sect. 5, using the real data collection obtained from Comodo Cloud Security Center, we systematically evaluate the effectiveness and efficiency of our proposed method in comparison with other alternate data mining approaches. Finally, Sect. 6 concludes.

2 Related Work

Signature-based method [4] is widely used in anti-malware industry for malware detection. However, malware attackers can easily evade this signature-based method through techniques such as encryption, packing, obfuscation, polymorphism, and metamorphism [5]. To gain profits, today's malware samples are created at a high speed (thousand per day). In order to remain effective, intelligent malware detection systems have been developed by applying data mining and machine learning techniques [1,2,15,16,25,26,29]. In these systems, the detection process is generally divided into two steps: *feature extraction* and *classification/clustering*. In the first step, various features, such as Windows Application Programming Interface (API) calls [29], program strings [15,20], and behavior based features [12], are extracted to capture the characteristics of the file samples. In the second step, classification or clustering techniques are used to automatically classify the file samples into different classes based on computational analysis of the feature representations. These intelligent malware detection systems are varied in their use of feature representations and classification/clustering methods. Most of such techniques simply treat the files as independent samples, however, the social relations among file samples may imply the inter-dependence among them and the usual ***i.i.d*** (independent and identical distributed) assumption may not hold for malware samples. As a result, ignoring the relations among file samples is a significant limitation of current malware detection methods.

Actually, besides file contents, the social relations between file samples (e.g., file co-existences, file co-operations) can provide invaluable information about the properties of file samples [27]. In recent years, limited research efforts have been conducted on file relation based malware detection. Chau et al. [7] applied a graph-based approach to infer the file reputations by analyzing file-to-machine

relations. Venzhega et al. [24] built regression classifier based on file placements for malware detection. In our previous work [27], we proposed a semi-parametric classification model for combining file content and file relations together for malware detection. In this paper, we will study how the relations between file samples can be used separately for malware detection. Different from the work in [8,9,22], we provide deep analysis of malware's social relation network and improve the graph inference algorithm for newly unknown malware detection.

3 Deep Analysis of Malware's Social Relation Network

In this section, we (1) first introduce the file relation graph construction, and (2) then provide deep analysis of malware's social relation network.

3.1 File Relation Graph Construction

Based on the collected file lists from the user clients, we construct a graph to describe the social relations among file samples (i.e., co-existence relationships). Generally, two files are related if they are shared by a group of clients (or equivalently, file lists). The file relation 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 user clients containing v_i and C_j be the set of user clients containing v_j. $|.|$ represents the size of a set. We define the *strength* of the relations (i.e., co-existence) between v_i and v_j based on the overlap between sets C_i and C_j, and use Jaccard similarity for measurement

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

If the strength $JS(v_i, v_j)$ between a pair of nodes (file samples) is greater than the specified threshold δ_{JS}, which indicates a *strong* (not *weak*) relation between v_i and v_j, then there is an edge between them. Each file is in a state of $S \in \{s_m, s_b, s_g\}$ (s_m: malicious, s_b: benign, s_g: unknown). The weight of edge between v_i and v_j, which is the probability of node v_i being in the state s_i and node v_j being in the state s_j, 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 user clients.

3.2 Graph Property Analysis

To analyze the property of malware's social relation network (co-existence relationship), we obtained a real dataset from Comodo Cloud Security Center: the dataset includes the file lists from 1,000 clients which describe file co-existence relations between 1,540 malware, 7,687 benign files and 2,250 unknown files (2,018 of them are analyzed by the anti-malware experts of Comodo Security Lab, 91 of them are malware and 1,927 of them are benign files). Figure 1 shows a zoomed-in view of a part of the constructed file relation graph (malware marked in red and benign files marked in green). From Fig. 1, we can see that many of the red nodes are associated with other red nodes and form some clusters, while the green nodes are also related to other green nodes and form their clusters. The nodes within the same cluster have strong relations with each other: (1) the red clusters may be the variants of malware families (e.g., family of online-game trojans); (2) the green clusters may be the related files of same applications (e.g., Acrobat installation archive and its related files).

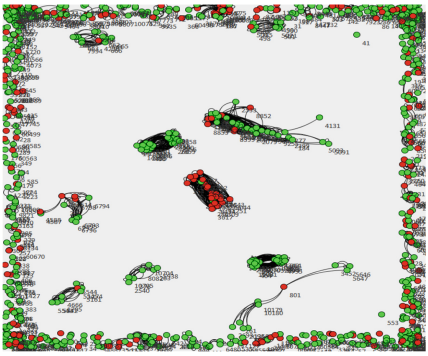


Fig. 1. Visualization of a social relation network among file samples (Color figure online)

Based on the dataset described above, we also further use fourteen measures in Table 1 to see the differences between benign file relation graph, ordinary malware (i.e., 1,220 malware whose the existence frequency is < 100) file relation graph and popular malware (i.e., 320 malware whose existence frequency is ≥ 100) file relation graph. In Table 1, from the comparisons of *G1* and *G2*, we can see that the measures of *components*, *component ratio*, *connectedness* and *fragmentation* are different between benign file relation graph and ordinary malware file relation graph; while from the comparisons of *G2* and *G3*, we can see that the measures of *avg degree*, *centralization* and *density* are different between ordinary malware file relation graph and top popular malware file relation graph. The different properties between benign file relation graph and malware file relation graph enable us to discriminate malware and benign files, while the different properties between ordinary malware file relation graph and popular malware file relation graph may allow us to predict the trend of malware prevalence.

Table 1. Graph property comparisons

NO.	Measures	G1	G2	G3
1	H-Index	129	125	125
2	Avg degree	49.842	***40.677***	***12.098***
3	Centralization	0.340	***0.344***	***0.081***
4	Density	0.018	***0.014***	***0.001***
5	Components	**103**	**11**	2
6	Component ratio	**0.036**	**0.004**	0.000
7	Connectedness	**0.964**	**0.996**	1.000
8	Fragmentation	**0.036**	**0.004**	0.000
9	Closure	0.081	0.091	0.047
10	Avg distance	2.744	3.056	3.408
11	SD distance	0.631	0.836	0.717
12	Diameter	5	7	4
13	Breadth	0.625	0.645	0.689
14	Compactness	0.375	0.355	0.311

"G1": graph constructed based on 7,687 benign files and files co-exist with them, "G2": graph constructed based on 1,220 ordinary malware and files co-exist with them, "G3": graph constructed based on 320 popular malware and files co-exist with them.

4 A Graph Inference Algorithm for Unknown File Labeling

Belief Propagation (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) [17,30]. It was first proposed by Pearl [19] to calculate marginal distribution in Markov Random Fields and Bayes Nets. 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 [18]. By definition, the message is the neighbor node's opinion about the current node's probability of being in the designated status. Mathematically, the message update equation in standard BP is

$$m_{i->j}(x_j) = \sum_{x_i \in S} f_{i->j}(x_i, x_j) g_i(x_i) \prod_{k \in N(i)/j} m_{k->i}(x_i), \tag{4}$$

where $m_{i->j}(x_j)$ is the message sent from node i to node j, that is, node i's belief that node j is in the state x_j; both $g_i(x_i)$ and $f_{i->j}(x_i, x_j)$ are typically called as energy functions, in which, $g_i(x_i)$ is the node potential, meaning the prior probability of node i being in the state x_i, while $f_{i->j}(x_i, x_j)$ is the edge potential, referring the probability of node i being in the state x_i and node j being in the state x_j; S is the set of states; $N(i)/j$ is the set of nodes neighboring node i (not including node j). BP algorithm stops when message updates

converge or a maximum number of iterations has finished. Then the belief value of each node is calculated as follow

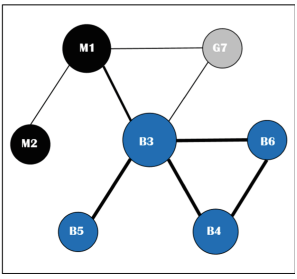
$$b_i(x_i) = k \times g_i(x_i) \prod_{k \in N(i)} m_{k->i}(x_i), \tag{5}$$

where k is a constant.

The standard BP is commonly called *sum-product* (from its message-update equation). A simple variant, called *max-product*, is used to estimate the state configuration with maximum probability, where the message update is the same as Eq. 4, except that sum is replaced by max, and the belief equation is the same as Eq. 5 [6].

Client ID	File List
C1	M1, M2
C2	M1, B3
C3	B3, B5
C4	B3, B4
C5	M1, B3, G7
C6	B3, B4, B6

(a) A sample datase



(b) file relation graph

Fig. 2. A sample dataset and its file relation graph constructed

The standard BP has been implemented in AESOP [22] for malware detection, however, it fails in our application. To put this into perspective, we use the sample dataset in Fig. 2(a) for further illustration, in which "M" denotes malware, "B" denotes benign file, and "G" is unknown file. Figure 2(b) is the constructed file relation graph based on the sample dataset (note that the weights of the nodes and edges are different). We employ the same energy functions designed in AESOP [22]. When the message updates converge (within threshold 10^{-3}), the belief values of the data nodes (i.e., BP_Belief) are shown in Table 2. From the results (i.e., BP_Class) in Table 2, we can see that file $B3$ and file $G6$ are misclassified.

In order to solve the problem above and make BP tailor to our application, we fine tune various components in BP and carefully design the message update and belief read-out functions. Before doing that, we first analyze the meaning of each energy function in our case for malware detection. In Eq. 4, $m_{i->j}(x_j)$, $f_{i->j}(x_i, x_j)$, and $g_i(x_i)$ represent message from node i to node j, edge potential, and node potential respectively. For malware detection problem, accordingly, $m_{i->j}(x_j)$ means the probability of node i believes that the neighbor node j being a benign file; $f_{i->j}(x_i, x_j)$ is the probability that node i and node j can be connected together; and $g_i(x_i)$ is the prior probability of node i being a

Table 2. The results of standard BP and IGIA based on Fig. 2

Nodes	BP_Belief	BP_Class	IGIA_Belief	IGIA_Class
M1	0.002642	M	0.008898	M
M2	0.005500	M	0.003617	M
B3	0.000006	M	0.121844	**B**
B4	0.207126	B	0.089217	B
B5	0.633993	B	0.067515	B
B6	0.207126	B	0.010625	B
G7	0.000007	M	0.078065	**B**

benign file. As described in Eq. 2, the weight of edge between a pair of nodes is the probability of node i being in the state x_i and node j being in the state x_j, which is the edge potential $f_{i->j}(x_i, x_j)$ in BP. Therefore, we fine tune and use the weight of edge $w(x_i, x_j)$ (defined in Eq. 2) between node i and j as the edge potential in our malware detection application. For node potential, $g_i(x_i)$ is the prior probability of node i being a benign file. We consider both its state and weight. Equation 6 shows our design of node potential in malware detection problem.

$$g_i(x_i) = \begin{cases} 0.5 + 0.5 * w(x_i) & \text{if } state(x_i) = s_b \\ 0.5 & \text{if } state(x_i) = s_g \\ 0.5 - 0.5 * w(x_i) & \text{if } state(x_i) = s_m, \end{cases} \tag{6}$$

where $w(x_i)$ is the weight of node i which can be calculated by Eq. 3 .

Instead of using sum-product, we redesign the message update equation as below

$$m_{i->j}(x_j) = \frac{1}{p} \sum_{x_i \in S} f_{i->j}(x_i, x_j) g_i(x_i) \frac{\sum_{k \in N(i)/j} m_{k->i}(x_i)}{q}, \tag{7}$$

where q equals to the number of the neighbors of node i (excluded node j), and p is a normalizing constant. In our application, we also initialize all the messages to 1. To be more robust to outliers, the belief read-out equation is redesigned as follow

$$b_i(x_i) = \frac{1}{r} g_i(x_i) Median_{k \in N(i)} \{m_{k->i}(x_i)\}, \tag{8}$$

where r is an adjustable constant.

The above Improved Graph Inference Algorithm is denoted as **IGIA**. Based on the energy functions as well as fine tuned message update and belief read-out equations designed in IGIA, using the same sample dataset in Fig. 2(a), the belief value of each node (i.e., IGIA_Belief) is shown in Table 2 and the results (i.e., IGIA_Class in Table 2) demonstrate that our proposed IGIA performs well in malware detection problem.

5 Experimental Results and Analysis

In this section, we conduct two set of experiments to evaluate our proposed graph inference algorithm for malware detection.

5.1 Experimental Setup

We use the same dataset obtained from Comodo Cloud Security Center as described in Sect. 3.2: the dataset includes the file lists from 1,000 clients which describe file co-existence relations between 1,540 malware, 7,687 benign files and 2,250 unknown files (2,018 of them are analyzed by the anti-malware experts of Comodo Security Lab, 91 of them are malware and 1,927 of them are benign files). The completely constructed file co-existence relation graph includes 11,477 nodes and 412,810 edges. The 1,540 malware and 7,687 benign files are used for training, while the 2,018 files labeled by human experts from the unknown sample collection are used for testing. We evaluate the malware detection performance of different methods using TP (true positive), TN (true negative), FP (false positive), FN (false negative), TPR (TP rate), FPR (FP rate), and ACY (accuracy).

5.2 Comparisons of Different Graph Inference Algorithms

In this section, we compare our proposed graph inference algorithm (IGIA) with standard BP (sum-product) implemented in AESOP [22] and BP with max-product. The results in Table 3 show that our proposed graph inference algorithm (IGIA) performs better than other two BP algorithms, due to our well designed energy functions and tuned message update as well as belief read-out.

Table 3. Comparisons of different graph inference algorithms

Training	TP	FP	TN	FN	ACY
IGIA	1,533	254	7,433	7	**0.9717**
Sum-product	1,469	7,404	283	71	0.1899
Max-product	1,244	6,142	1,545	296	0.3023
Testing	TP	FP	TN	FN	ACY
IGIA	65	55	1,872	26	**0.9598**
Sum-product	89	1,812	115	2	0.1011
Max-product	70	1,507	420	21	0.2428

5.3 Comparisons with Other Classification Approaches

In this section, we compare the malware detection effectiveness and efficiency of our proposed graph inference algorithm (IGIA) and other classification approaches (e.g., Support Vector Machine (SVM) and Decision Tree (DT)). Table 4 shows that the our proposed graph inference algorithm (IGIA) outperforms the other typical classification methods in malware detection. For malware detection efficiency, based on the 2,018 testing samples, Fig. 3 also demonstrates that the proposed graph inference algorithm (IGIA) perform better than the other two classifiers.

Table 4. Comparisons of malware detection effectiveness between IGIA and other classification methods

Training	TP	FP	TN	FN	ACY
IGIA	1,533	254	7,433	7	**0.9717**
SVM	926	158	7,529	614	0.9163
DT	1,021	708	6,979	519	0.8670
Testing	TP	FP	TN	FN	ACY
IGIA	65	55	1,872	26	**0.9598**
SVM	54	232	1,695	37	0.8667
DT	57	297	1,630	34	0.8360

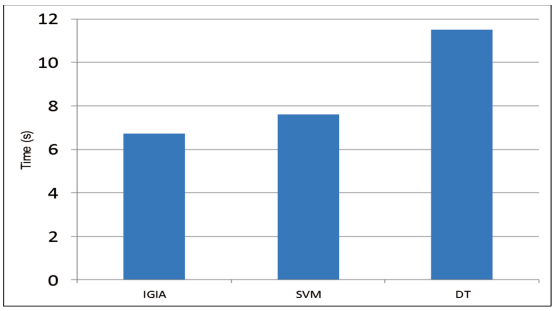


Fig. 3. Comparisons of malware detection efficiency between the proposed graph inference algorithm (IGIA) and other classification methods

6 Conclusion

In this paper, we provide deep analysis of malware's social relation network and study how it can be used for malware detection; we also propose an effective

graph inference algorithm (IGIA) for malware detection based on the constructed file relation graphs. Empirical studies on large and real data collection obtained from Comodo Cloud Security Center illustrate that our proposed method outperforms other alternate data mining based approaches in malware detection. In our future work, we will further investigate more social relation based features (e.g., file co-operations) for malware detection and analyze the properties of different kinds of social relations among file samples. We will also design a full detection solution by combining social 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 helpful discussions and supports. The work of S. Hou, Lingwei Chen, and Y. Ye is supported by the U.S. National Science Foundation under grant CNS-1618629; the work of Lifei Chen is supported by the Chinese National Science Foundation under grant 61672157.

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). https://doi.org/10.1007/978-3-540-74320-0_10
2. Bayer, U., Comparetti, P.M., Hlauschek, C., Kruegel, C., Kirda, E.: Scalable, behavior-based malware clustering. In: NDSS 2009 (2009)
3. Bayer, U., Moser, A., Kruegel, C., Kirda, E.: Dynamic analysis of malicious code. J. Comput. Virol. **2**(1), 67–77 (2006)
4. Beaucamps, P., Filiol, E.: Malware pattern scanning schemes secure against black box analysis. J. Comput. Virol. **2**(1), 35–50 (2006)
5. Beaucamps, P., Filiol, E.: On the possibility of practically obfuscating programs towards a unified perspective of code protection. J. Comput. Virol. **3**(1), 3–21 (2007)
6. Bishop, C.: Pattern Recognition and Machine Learning. Information Science and Statistics. Springer, New York (2006)
7. Chau, D.H., Nachenberg, C., Wilhelm, J., Wright, A., Faloutsos, C.: Polonium: tera-scale graph mining for malware detection. In: SIAM International Conference on Data Mining (SDM) (2011)
8. Chen, L., Hardy, W., Ye, Y., Li, T.: Analyzing file-to-file relation network in malware detection. In: Wang, J., Cellary, W., Wang, D., Wang, H., Chen, S.-C., Li, T., Zhang, Y. (eds.) WISE 2015. LNCS, vol. 9418, pp. 415–430. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-26190-4_28
9. Chen, L., Li, T., Abdulhayoglu, M., Ye, Y.: Intelligent malware detection based on file relation graphs. In: Proceedings of the 9th IEEE International Conference on Semantic Computing, pp. 85–92 (2015)
10. Clarkson, K.L.: Large-scale malware analysis, detection, and signature generation. Ph.D. dissertation, University of Michigan (2011)
11. CSI: 12th annual edition of the CSI computer crime and security survey. Technical report, Computer Security Institute (2007)
12. Filiol, E., Jacob, G., Liard, M.: Evaluation methodology and theoretical model for antiviral behavioural detection strategies. J. Comput. Virol. **3**(1), 27–37 (2007)

13. Jung, W., Kim, S., Choi, S.: Poster: deep learning for zero-day flash malware detection. In: S&P 2015 (2015)
14. 2013–2014 Internet security report in China (2014). http://www.ijinshan.com/news/2014011401.shtml
15. Kolter, J.Z., Maloof, M.A.: Learning to detect malicious executables in the wild. In: KDD 2004, pp. 470–478 (2004)
16. Li, Y., Ma, R., Jiao, R.: A hybrid malicious code detection method based on deep learning. IJSIA **9**, 205–216 (2015)
17. McGlohon, M., Bay, S., Anderle, M.G., Steier, D.M., Faloutsos, C.: Snare: a link analytic system for graph labeling and risk detection. In: KDD 2009 (2009)
18. 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)
19. Pearl, J.: Reverend bayes on inference engines: a distributed hierarchical approach. In: Proceedings of the Second National Conference on Artificial Intelligence, pp. 133–136 (1982)
20. Reddy, D.K.S., Pujari, A.K.: N-gram analysis for computer virus detection. J. Comput. Virol. **2**(3), 231–239 (2006)
21. Shah, S., Jani, H., Shetty, S., Bhowmick, K.: Virus detection using artificial neural networks. Int. J. Comput. Appl. **84**, 17–23 (2013)
22. Tamersoy, A., Roundy, K., Chau, D.H.: Guilt by association: large scale malware detection by mining file-relation graphs. In: KDD 2014 (2014)
23. Zeus: a persistent criminal enterprise (2010). http://www.trendmicro.com/cloudcontent/us/pdfs/security-intelligence/white-papers/wpzeuspersistent-criminal-enterprise.pdf
24. Venzhega, A., Zhinalieva, P., Suboch, N.: Graph-based malware distributors detection. In: WWW 2013 (2013)
25. Wang, J., Deng, P., Fan, Y., Jaw, L., Liu, Y.: Virus detection using data mining techniques. In: ICDM (2003)
26. Ye, Y., Li, T., Chen, Y., Jiang, Q.: Automatic malware categorization using cluster ensemble. In: KDD 2010, pp. 95–104 (2010)
27. 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: KDD 2011, pp. 222–230 (2011)
28. Ye, Y., Li, T., Jiang, Q., Han, Z., Wan, L.: Intelligent file scoring system for malware detection from the gray list. In: KDD 2009 (2009)
29. Ye, Y., Wang, D., Ye, D.: IMDS: intelligent malware detection system. In: KDD 2007, pp. 1043–1047 (2007)
30. Yedidia, J.S., Freeman, W.T., Weiss, Y.: Understanding belief propagation and its generalizations. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2003)