File Relation Graph Based Malware Detection Using Label Propagation

Ming Ni[1](✉), Qianmu Li[1], Hong Zhang[1], Tao Li[2], and Jun Hou[3]

[1] School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China
nq1027@gmail.com

[2] School of Computer Science and Technology, School of Software, Nanjing University of Posts and Telecommunications, Nanjing 210046, China

[3] School of Humanities and Social Sciences, Nanjing University of Science and Technology, Nanjing 210094, China

Abstract. The rapid development of malicious software programs has posed severe threats to Computer and Internet security. Therefore, it motivates anti-malware industry to develop novel methods which are capable of protecting users against new threats. Existing malware detectors mostly treat the file samples separately using supervised learning algorithms. However, ignoring of relationship among file samples limits the capability of malware detectors. In this paper, we present a new malware detection method based on file relation graph to detect newly developed malware samples. When constructing file relation graph, k-nearest neighbors are chosen as adjacent nodes for each file node. Files are connected with edges which represent the similarity between the corresponding nodes. Label propagation algorithm, which propagates label information from labeled file samples to unlabeled files, is used to learn the probability that one unknown file is classified as malicious or benign. We evaluate the effectiveness of our proposed method on a real and large dataset. Experimental results demonstrate that the accuracy of our method outperforms other existing detection approaches in classifying file samples.

Keywords: Malware detection · File relation graph · kNN · Label propagation

1 Introduction

With the rapid development of Computer and Internet technology, computer security becomes more and more prevalent over past decades. Malware (short for **mal**icious soft**ware**), including Viruses, Backdoors, Spyware, Trojans, Worms and Botnets, is software that spread and infect computers for malicious intent of an attacker [5]. In the form of executable code, scripts, active content, and other softwares, malware samples can be used to disrupt computer operation, gather sensitive information, or gain access to private computer systems, and may cause serious damages and financial losses to computers and users. Malware detection is thus becoming more and more important due to its damage to the security and the economic loss of people.

Currently, the main approach of protecting against malware is signature-based method which is widely adopted by most anti-malware companies [6,7]. Signature is a particular piece of code which is obtained after being analyzed manually by computer security experts and expressed in the form of byte or instruction sequences and is unique for each known malware [8]. However, due to the rapid development of malware techniques, a huge number of malware samples are being generated or mutated every day. Meanwhile, malware writers have employed advanced development toolkit, including encryption, polymorphism, and metamorphism to make malware samples be immune to signature-based detection. It poses a big threat to signature-based detection. Human experts cannot analyze each new file manually, and the required responding time is limited. This issue has motivated anti-malware industry to redesign their security systems for detecting malware samples. Recently, many research efforts have been conducted on malware detection using data mining techniques. Researchers have shifted from traditional signature-based method to file-content-analysis based approaches to detect and classify malware with static or dynamic features [1,2,10–13,19,21,22]. These techniques applied data mining algorithms for malware detection based on content features, such as instructions, control flow extracted from binary codes and API call sequences tracked from runtime environments.

In this paper, instead of using the content information of file samples, we investigate how file relations can be used to detect malware samples and employ a Label Propagation method for classifying file samples based on the constructed file relation graphs. A real and large scale file relation dataset from an anti-malware industry company is used in the experiments. The scale of this dataset is representative including 69,165 file samples (3,095 malware, 22,583 benign files, and 43,487 unknown files) on 3,793 clients.

This paper makes the following contributions:

1. Unlike classic classifiers based on only the file content information, we make use of the relationships among file samples and apply graph mining algorithm for malware detection. Relations with other known files are used to identify the unknown file samples.
2. We use the k-nearest neighbors of each file sample to construct a file relation graph for inferring each file's probability of being malicious or benign.
3. A label propagation algorithm is used to propagate the label information from labeled files to unlabeled files.
4. The empirical evaluation on a real and large data collection from an anti-malware industry company is performed and demonstrates the performance of our method.

The remainder of this paper is organized as follows. Section 2 presents the background and discuss the related work. Details of the dataset is described in Sect. 3. We discuss how file relations and the Label Propagation algorithm can be used to perform malware detection in Sect. 4. Experiments are conducted to evaluate the effectiveness and efficiency of our proposed method by comparing with the baselines in Sect. 5. Finally, we state the conclusions and future studies in Sect. 6.

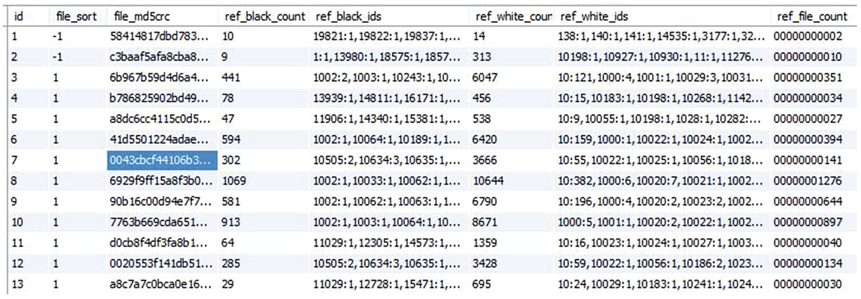
2 Background and Related Work

In recent years, an increasing number of studies have been conducted on developing efficient algorithms to detect malware by data mining and machine learning techniques [1,8,9,13,16–19,21,22]. In [8], Jeffrey et al. developed a statistical method to extract virus signatures automatically, it is the first major work applying data mining techniques to detect malware. Schultz et al. [12] used DLL information, strings and n-grams to train RIPPER, Naive Bayes and Multi Naive Bayes to classify malware. Assaleh et al. [1] created class profiles of various lengths according to the number of most frequent n-grams within the class with different n-gram sizes. Kolter et al. [9] selected the most relevant n-grams on 1971 benign and 1651 malicious file samples, then different classification methods including Naive Bayes, Support Vector Machine (SVM), and Decision Tree (DT), were compared based on these n-grams for malware detection. In [21], Ye et al. developed an Intelligent Malware Detection System (IMDS) which uses Objective-Oriented Association classification based on Windows API call sequences. An OOA Fast FP-Growth algorithm was developed. Their experiments showed that OOA-based method outperforms the Apriori algorithm for association rule generation.

The aforementioned methods are all based on the file contents, including Application Programming Interface calls and program code strings. Besides file contents, relations among file samples can also be used to extract invaluable information about the properties of file samples. In recent years, some research efforts have been conducted on detecting malware based on file relation graphs [3,4,14,15,20]. Chau et al. [3] presented a novel method based on Belief Propagation algorithm to infer file reputation using file-machine relations. In [20], Ye et al. built a semi-parametric classifier model that combines file-to-file relationship with file contents information for malware detection. Tamersoy et al. proposed AESOP, a scalable algorithm, which leverages locality-sensitive hashing to measure similarity between files and employs a tuned BP algorithm on the file-bucket graph based on LSH [14].

3 Data Description

In this section, we describe the dataset used in our work. We obtain the dataset from an anti-malware industry which contains 69,165 file samples (3,095 malware, 22,583 benign files, and 43,487 unknown files) and relations between these file samples [20]. Figure 1 shows the structure of the file relation database including 8 fields: file id, file label ("1" is for benign file, "-1" denotes malicious file, and "0" represents unknown file), file name, number of malware that the file co-exists, malware ids that the file co-exists, the number of benign files that the file co-exists, benign file ids that the file co-exists, number of clients in which the file exists.



id	file_sort	file_md5crc	ref_black_count	ref_black_ids	ref_white_cour	ref_white_ids	ref_file_count
1	-1	58414817dbd783...	10	19821:1,19822:1,19837:1,...	14	138:1,140:1,141:1,14535:1,3177:1,32...	0000000002
2	-1	c3baaf5afa8cba8...	9	1:1,13980:1,18575:1,1857...	313	10198:1,10927:1,10930:1,11:1,11276...	0000000010
3	1	6b967b59d4d6a4...	441	1002:2,1003:1,10243:1,10...	6047	10:121,1000:4,1001:1,10029:3,10031...	0000000351
4	1	b786825902bd49...	78	13939:1,14811:1,16171:1,...	456	10:15,10183:1,10198:1,10268:1,1142...	0000000034
5	1	a8dc6cc4115c0d5...	47	11906:1,14340:1,15381:1,...	538	10:9,10055:1,10198:1,1028:1,10282:...	0000000027
6	1	41d5501224adae...	594	1002:1,10064:1,10189:1,1...	6420	10:159,1000:1,10022:1,10024:1,1002...	0000000394
7	1	0043cbcf44106b3...	302	10505:2,10634:3,10635:1,...	3666	10:55,10022:1,10025:1,10056:1,1018...	0000000141
8	1	6929f9ff15a8f3b0...	1069	1002:1,10033:1,10062:1,1...	10644	10:382,1000:6,10020:7,10021:1,1002...	0000001276
9	1	90b16c00d94e7f7...	581	1002:1,10062:1,10063:1,1...	6790	10:196,1000:4,10020:2,10023:2,1002...	0000000644
10	1	7763b669cda651...	913	1002:1,1003:1,10064:1,10...	8671	1000:5,1001:1,10020:2,10022:1,1002...	0000000897
11	1	d0cb8f4df3fa8b1...	64	11029:1,12305:1,14573:1,...	1359	10:16,10023:1,10024:1,10027:1,1003...	0000000040
12	1	0020553f141db51...	285	10505:2,10634:3,10635:1,...	3428	10:59,10022:1,10056:1,10186:2,1023...	0000000134
13	1	a8c7a7c0bca0e16...	29	11029:1,12728:1,15471:1,...	695	10:24,10029:1,10183:1,10241:1,1024...	0000000030

Fig. 1. Sample File Relation Database

Usually, the number of benign files is much larger than that of malware samples. It leads to the imbalanced data distribution which can be seen from the dataset. Both the number and the relations are imbalanced. Figures 2 and 3 show the distribution of the co-occurrence between malware samples and the co-occurrence between malware samples and benign files respectively.

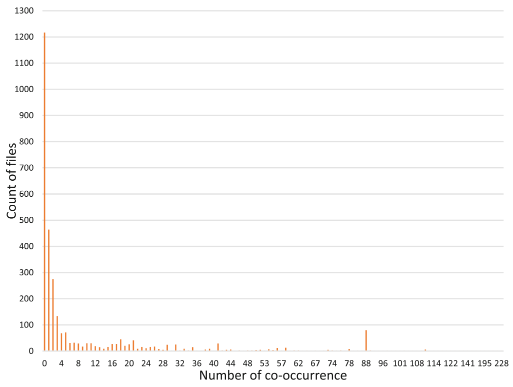


Fig. 2. Co-occurrence between Malware Samples

Note that the file lists were collected from users' clients. It is unnecessary and unpractical for the clients to collect all the file samples from users' machines, the clients only submit the suspicious file samples to the server for further analysis. So that, only the associations with labeled file samples are recorded in the database, the relationship between unknown file samples is missing. We will use the co-occurrence information for each unknown file sample to calculate the similarity between unknown file samples, which will be described in next section.

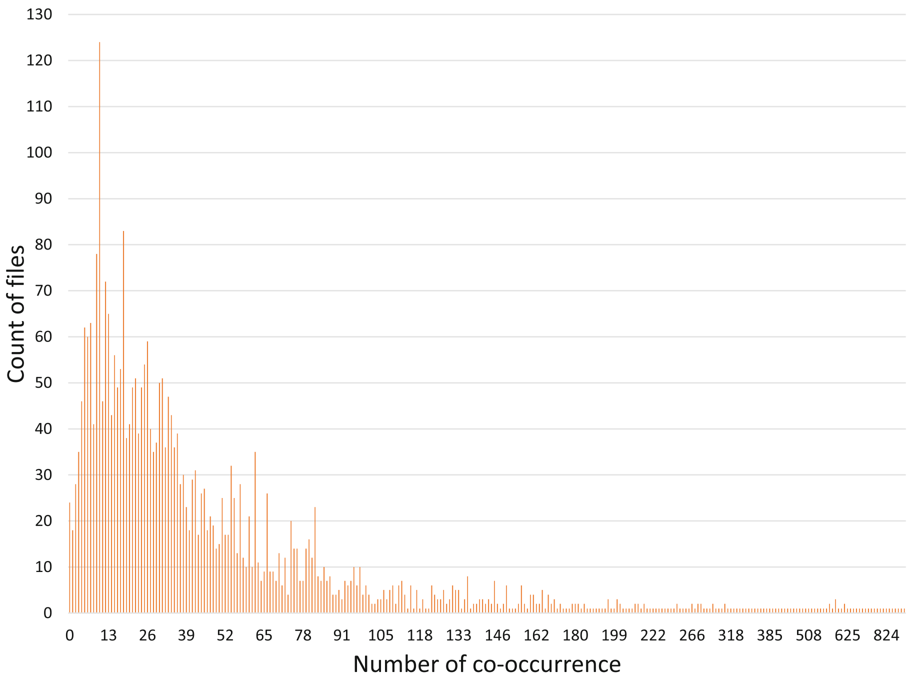


Fig. 3. Co-occurrence between Malware Samples and Benign Files

4 Graph-Based Malware Detection Using Label Propagation

4.1 File Relation Graph Construction

Based on the structure property of the dataset described in Sect. 3, an undirected weighted graph is constructed to represent the relations among file samples. The graph is defined as $G = (V, E, W)$, where V is set of nodes corresponding to the file samples, E represents the relations among the nodes, and W corresponds to the weights of each edge. Here, we define the similarity between file f_i and f_j as the co-occurrence strength. Let C_i and C_j denote the set of clients in which the file f_i and f_j exists respectively. The Jaccard similarity measure is used to calculate the co-occurrence strength as follows.

$$sim(f_i, f_j) = \frac{|C_i \cap C_j|}{|C_i \cup C_j|}, \tag{1}$$

where $|C|$ is the size of set C. The value of this measure is between 0 and 1; "0" indicates no co-occurrence relationship, "1" indicates a full co-occurrence relationship.

As mentioned earlier, the relationship between unknown file samples is missing. Here, we use the relationship with labeled file samples of each unknown file to measure the similarity between unknown file samples. Let M_i represent the set of file samples which co-exist with unknown file sample f_i. The similarity between two unknown file samples f_i and f_j is:

$$sim(f_i, f_j) = \sum_{m \in M_i \cap M_j} sim(m, i) * sim(m, j). \quad (2)$$

In order to filter out the noisy data, we choose the k-nearest neighbors of each file by applying the kNN based method. If file f_i is in k-nearest neighbor of file f_j, then there is an edge between them. The weight of the edge is the similarity between file f_i and f_j.

To further illustrate, a file relation dataset sample is given as Table 1, in which 6(3) means that the file co-exists with file No.6 in three clients. Based on the given relations, an undirected weighted graph is constructed as shown in Fig. 4. The figure shows the relations among the files in the case of $k = 3$, and the number on each edge indicates the weight. The key idea of our problem can be described as: An unlabeled file sample can be labeled as malware or benign based on their co-occurrence with labeled files.

Table 1. File Relation Dataset Sample

ID	Label	Co-exists with Malware	Co-exists with Benign File	Count of Clients
1	0	7(1),8(1),10(1)	6(1)	2
2	0	4(1)	3(2),6(2)	2
3	1	4(1),8(1)	5(2),6(3)	4
4	−1	8(1),10(1)	3(1),6(1)	2
5	1	8(1)	3(2),6(1)	2
6	1	4(1),8(1)	3(3),5(1)	4
7	−1	10(1)	——	1
8	−1	4(1),10(1)	3(1),5(1),6(1)	3
9	0	——	3(1),5(1),6(1)	1
10	−1	4(1),7(1),8(1)	——	2

4.2 Label Propagation

Label Propagation is a graph-based semi-supervised learning method, which lets every labeled data spread its label information to the whole graph until all unlabeled data have a stable label states [23].

Let $(x_1, y_1)...(x_l, y_l)$ denote labeled data, where $y_1...y_l$ are class labels, and $(x_{l+1}, y_{l+1})...(x_{l+u}, y_{l+u})$ be unlabeled data. The key idea of label propagation is that data points with high similarity tend to have the same labels. Label

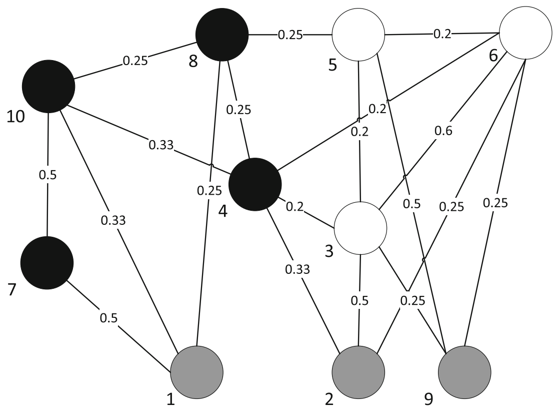


Fig. 4. Constructed Graph based on Table 1

information of labeled nodes need to be propagated to all nodes through the edges.

Define Y as a $(l+u) \times C$ label matrix, where Y_{ij} represents the probability of node x_i being labeled as y_j and C is the number of classes. In other words, Y represents the label probability distribution of each node. T is a probabilistic transition matrix defined as

$$T_{ij} = P(j \to i) = \frac{w_{ij}}{\sum_{k=1}^{l+u} w_{ij}}, \tag{3}$$

where T_{ij} denotes the probability of jumping from node j to i. Algorithm 1 presents the method proposed by Zhu in [23].

Algorithm 1. Label Propagation

```
1. Initialization. Set $Y$ be the initial labels attached to each node, where $Y_{ij} = 1$ if
   $x_i$ is labeled as $y_j$.
repeat
    (1). Propagate labels of any node to its neighbors by $Y \leftarrow \overline{T}Y$, where $\overline{T}$ is row-
    normalized matrix of $T$, i.e. $\overline{T}_{ij} = T_{ij}/\sum_k T_{ik}$.
    (2). Clamp the labeled data.
until $Y$ converges
2. Assign $x_i$ with a label using $y_i = \arg\max_j Y_{ij}$.
```

Define Y_L as the top l rows of Y which are labeled data and Y_U as the remaining u rows standing for unlabeled data. Due to the clamping operation, Y_L never changes, so we only need to focus on Y_U. It is shown that the algorithm converges to a unique fixed point and the solution is $Y_U = (I - \overline{T}_{uu})^{-1}\overline{T}_{ul}Y_L$. Here, $\overline{T}_{uu}$ and $\overline{T}_{ul}$ are sub-matrices obtained by splitting $\overline{T}$ after the l-th row and the l-th column.

4.3 Malware Detection Using Label Propagation

Based on the constructed file relation graph as well as the label propagation algorithm described in previous subsection, the whole process of our proposed method is described in Algorithm 2.

Algorithm 2. Algorithm for malware detection

```
Input: Raw file lists data
Output: Class label of each file sample
  1. Calculate the similarity for each pair of associated files;
  2. Calculate the similarity for each pair of unlabeled files based on their co-occurrence;
  3. Choose $k$ nearest neighbors for each file as neighbors in the graph;
  4. Initialize graph $G = (V, E, W)$;
  5. Perform the Label Propagation algorithm described in Algorithm 1;
  6. Assign labels(i.e., malicious or benign) to unlabeled file samples.
```

5 Experiment

In this section, we conduct two sets of experiments: (1) In the first set of experiments, we evaluate the effectiveness of kNN method applied for neighbor selection and choose the best value of k for the rest experiments. (2) In the second set of experiments, we evaluate the effectiveness of our proposed method for malware detection by comparing with baseline methods. All algorithms are evaluated with the dataset described in Sect. 3.

5.1 Experiments Setting

All algorithms in following subsections are implemented on a laptop of Windows 8 OS with Intel Core i7 2.7 GHz Duo CPU and 8 GB RAM using JAVA 1.7. The evaluation metrics are described below.

- **True Positive(TP)**: Number of samples labeled as malicious correctly.
- **True Negative(TN)**: Number of samples labeled as benign correctly.
- **False Positive(FP)**: Number of samples labeled as malicious incorrectly.
- **False Negative(FN)**: Number of samples labeled as benign incorrectly.
- **TP Rate(TPR)**: $\frac{TP}{TP+FN}$
- **FP Rate(FPR)**: $\frac{FP}{TN+FP}$
- **Accuracy(ACC)**: $\frac{TP+TN}{TP+TN+FP+FN}$

5.2 Performance Evaluation of Neighbor Selection Using kNN

When constructing the file relation graph, we choose the k-nearest neighbors of each file samples to filter out the noisy data and keep the most similar neighbors. In this section, we evaluate the effectiveness of applying kNN method. We run the algorithm without applying kNN method, and 5 times with $k = 10$, $k = 30$, $k = 50$, $k = 70$ and $k = 100$ respectively. From Table 2 and Fig. 5, we saw an improvement of about 13 % on TP (True Positive) and 10 % on TN (True Negative) after applying kNN by setting $k = 50$.

Table 2. Effectiveness of Applying kNN

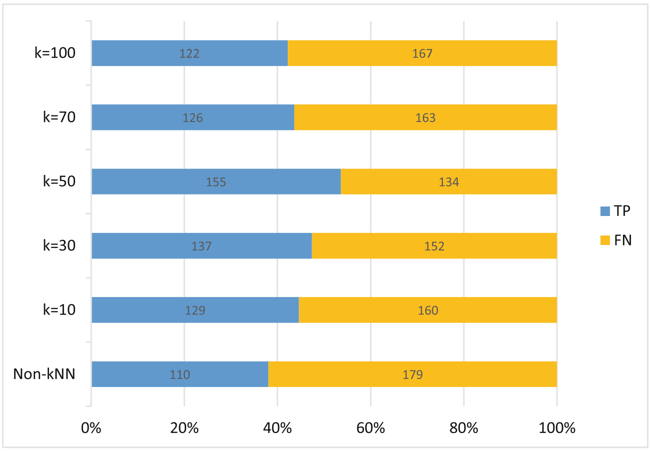
Method	TP	FP	TN	FN	ACC
Non-kNN	110	301	3,396	179	0.8796
$k = 10$	109	224	3,473	180	0.8986
$k = 30$	113	179	3,518	176	0.9109
$k = 50$	155	67	3,630	134	0.9496
$k = 70$	126	139	3,558	163	0.9242
$k = 100$	122	199	3,498	167	0.9082

5.3 Comparisons of Label Propagation with Other Methods

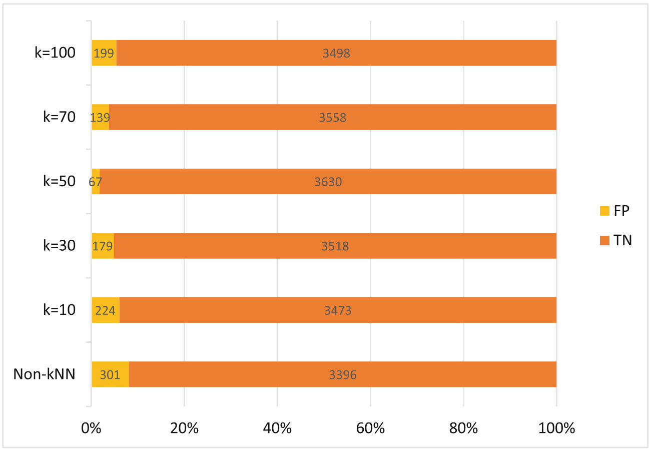
In this subsection, we compare the effectiveness of our proposed method with other methods including both graph-based and content-based classification approaches. Four baseline methods were compared: AESOP in [14], Malware Distributor Detector (MDD) in [15], Support Vector Machine (SVM), and Random Forest (RF).

Cross Validation: We use 10-fold cross validation scheme to evaluate the performance of our proposed method. At each round, we set the labels of files in the test set to 0 and the probabilities of being malicious and benign both to 0.5. For each fold, we run our proposed algorithm with baseline methods and record the ACC (Accuracy). Quantitative results on the 10-fold validation are shown in Fig. 6. The notch marks the 95 % confidence interval for the medians. The figure demonstrates that our proposed method makes an significant improvement on accuracy compared to the best performance among the baseline methods, with $k = 50$.

Prediction: 3,986 files with ground truth (includes 289 malware, 3,697 benign files) were selected at random as the test data for evaluation, and the rest data were used as the training data. Here we set $k = 50$. Table 3 and Fig. 7 present the



(a) TP Versus FN



(b) TN Versus FP

Fig. 5. Effectiveness of Applying kNN

results of our proposed method along with the four baseline methods. The comparison results illustrate that our proposed algorithm outperforms other methods in malware detection on the large and real datasets.

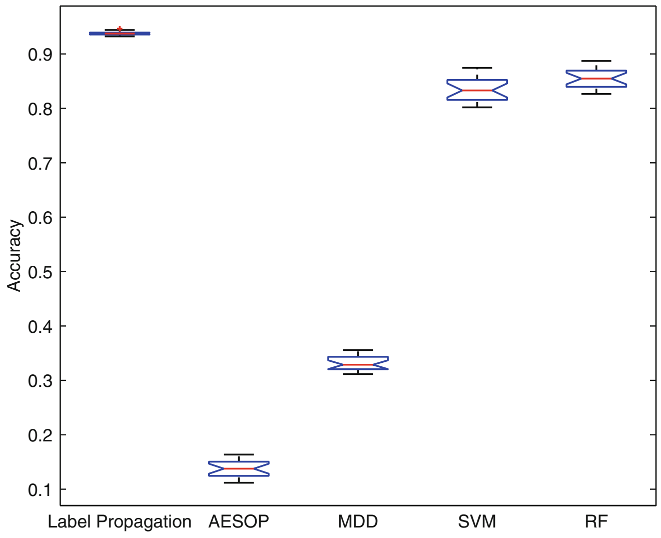


Fig. 6. Comparisons of Accuracy for 10-fold validation

Table 3. Quantitative comparisons of our proposed method with baseline methods on large and real data

Method	TP	FP	TN	FN	ACC
Label Propagation	155	67	3,630	134	0.9496
AESOP	198	3,385	312	91	0.1279
MDD	98	2,493	1,204	191	0.3266
SVM	81	461	3,236	208	0.8321
RF	69	398	3,299	220	0.8449

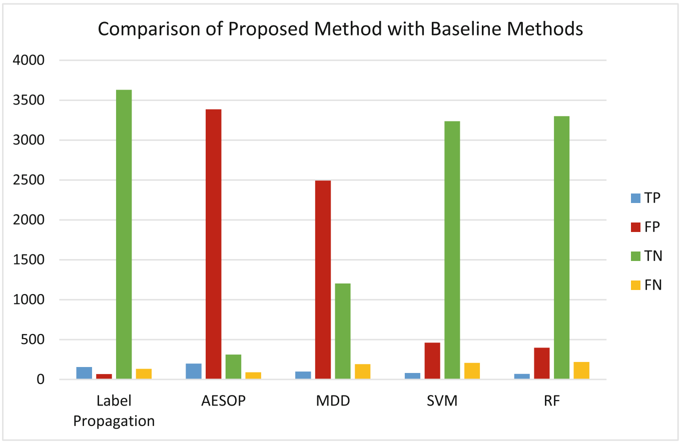


Fig. 7. Comparisons of Proposed Method with Baseline Methods

6 Conclusion and Future Work

In this paper, we study how to use file relations for malware detection. The associations between file samples are used to compute the file similarity values to construct file-relation graph by kNN method. A Label Propagation algorithm is applied for classifying file samples based on the constructed file-relation graph. We use a real and large dataset consisting of file co-occurrence records from users' clients. Comprehensive experiments are performed to compare our proposed method with other existing malware detection approaches. The experimental results demonstrate that the accuracy of our proposed method outperform other malware detection methods using data mining techniques. For the future work, we plan to further explore the combination of file relation information and file contents to reduce the false positive and negative rates.

Acknowledgment. This paper is supported by the National Natural Science Foundation of China under Grant No. 61272419, the Future Network Science Foundation of Jiangsu under Grant No. BY2013095-3-02, the Prospective research project in Jiangsu Province under Grant No. BY2014089, and the Lianyungang Scientific and Technological Project under Grant No. CH1304.

References

1. Abou-Assaleh, T., Cercone, N., Keselj, V., Sweidan, R.: N-gram-based detection of new malicious code. In: Proceedings of the 28th Annual International Computer Software and Applications Conference, COMPSAC 2004, vol. 2, pp. 41–42. IEEE (2004)
2. 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)
3. Chau, D.H., Nachenberg, C., Wilhelm, J., Wright, A., Faloutsos, C.: Polonium: tera-scale graph mining and inference for malware detection. In: SIAM International Conference on Data Mining, vol. 2 (2011)
4. Chen, l., Li, T., Abdulhayoglu, M., Ye, Y.: Intelligent malware detection based on file relation graphs. In: 2015 IEEE International Conference on Semantic Computing (ICSC), pp. 85–92. IEEE (2015)
5. Egele, M., Scholte, T., Kirda, E., Kruegel, C.: A survey on automated dynamic malware-analysis techniques and tools. ACM Comput. Surv. (CSUR) **44**(2), 6 (2012)
6. Filiol, E.: Malware pattern scanning schemes secure against black-box analysis. J. Comput. Virol. **2**(1), 35–50 (2006)
7. Filiol, E., Jacob, G., Le Liard, M.: Evaluation methodology and theoretical model for antiviral behavioural detection strategies. J. Comput. Virol. **3**(1), 23–37 (2007)
8. Kephart, J.O., Arnold, W.C.: Automatic extraction of computer virus signatures. In: 4th Virus Bulletin International Conference, pp. 178–184 (1994)
9. Kolter, J.Z., Maloof, M.A.: Learning to detect malicious executables in the wild. In: Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 470–478. ACM (2004)

10. 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 Trans. Manag. Inf. Syst. (TMIS) **2**(3), 16 (2011)
11. Reddy, D.K.S., Pujari, A.K.: N-gram analysis for computer virus detection. J. Comput. Virol. **2**(3), 231–239 (2006)
12. Schultz, M.G., Eskin, E., Zadok, E., Stolfo, S.J.: Data mining methods for detection of new malicious executables. In: Proceedings of the 2001 IEEE Symposium on Security and Privacy, S&P 2001, pp 38–49. IEEE (2001)
13. Siddiqui, M., Wang, M.C., Lee, J.: A survey of data mining techniques for malware detection using file features. In: Proceedings of the 46th Annual Southeast Regional Conference on XX, pp. 509–510. ACM (2008)
14. Tamersoy, A., Roundy, K., Chau, D.H.: Guilt by association: large scale malware detection by mining file-relation graphs. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1524–1533. ACM (2014)
15. Venzhega, A., Zhinalieva, P., Suboch, N.: Graph-based malware distributors detection. In: Proceedings of the 22nd International Conference on World Wide Web Companion, pp. 1141–1144. International World Wide Web Conferences Steering Committee (2013)
16. Ye, Y., Chen, L., Wang, D., Li, T., Jiang, Q., Zhao, M.: SBMDS: an interpretable string based malware detection system using SVM ensemble with bagging. J. Comput. Virol. **5**(4), 283–293 (2009)
17. Ye, Y., Li, T., Huang, K., Jiang, Q., Chen, Y.: Hierarchical associative classifier (HAC) for malware detection from the large and imbalanced gray list. J. Intell. Inf. Syst. **35**(1), 1–20 (2010)
18. Ye, Y., Li, T., Jiang, Q., Han, Z., Wan, L.: Intelligent file scoring system for malware detection from the gray list. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1385–1394. ACM (2009)
19. Ye, Y., Li, T., Jiang, Q., Wang, Y.: CIMDS: adapting postprocessing techniques of associative classification for malware detection. IEEE Trans. Syst. Man Cybern. Part C: Appl. Rev. **40**(3), 298–307 (2010)
20. 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 International Conference on Knowledge Discovery and Data Mining, pp. 222–230. ACM (2011)
21. Ye, Y., Wang, D., Li, T., Ye, D.: IMDS: Intelligent malware detection system. In: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1043–1047. ACM (2007)
22. Ye, Y., Wang, D., Li, T., Ye, D., Jiang, Q.: An intelligent pe-malware detection system based on association mining. J. Comput. Virol. **4**(4), 323–334 (2008)
23. Zhu, X., Ghahramani, Z.: Learning from labeled and unlabeled data with label propagation. Technical report. Citeseer (2002)