Mal-Netminer: Malware Classification based on Social Network Analysis of Call Graph

Jae-wook Jang
Graduate School of Information Security
Korea University
Seoul, Republic of Korea
changkr@korea.ac.kr

Jiyoung Woo
Graduate School of Information Security
Korea University
Seoul, Republic of Korea
jywoo@korea.ac.kr

Jaesung Yun
Graduate School of Information Security
Korea University
Seoul, Republic of Korea
yjs8888@korea.ac.kr

Huy Kang Kim*
Graduate School of Information Security
Korea University
Seoul, Republic of Korea
cenda@korea.ac.kr

ABSTRACT

In this work, we aim to classify malware using automatic classifiers by employing graph metrics commonly used in social network analysis. First, we make a malicious system call dictionary that consists of system calls found in malware. To analyze the general structural information of malware and measure the influence of system calls found in malware, we adopt social network analysis. Thus, we use social network metrics such as the degree distribution, degree centrality, and average distance, which are implicitly equivalent to distinct behavioral characteristics. Our experiments demonstrate that the proposed system performs well in classifying malware families within each malware class with accuracy greater than 98%. As exploiting the social network properties of system calls found in malware, our proposed method can not only classify the malware with fewer features than previous methods adopting graph features but also enables us to build a quick and simple detection system against malware.

Categories and Subject Descriptors

C.2.0 [**COMPUTER-COMMUNICATION NETWORKS**]: General—*Security and protection*

Keywords

Social network analysis (SNA), Degree distribution, System call graph, Dynamic analysis, Malware

*Corresponding author.

WWW'14 Companion, April 7–11, 2014, Seoul, Korea.
ACM 978-1-4503-2745-9/14/04.
http://dx.doi.org/10.1145/2567948.2579364.

1. INTRODUCTION

Given the prevalence of the malware threat, many previous studies have proposed various malware analysis methodologies. The proposed malware classification or detection methods exploit a system or API call set, an instruction pattern set, or call graph matching. Since such malicious call or instruction sets form a specific frequency distribution or have unique call sequences, they can be used as one of the metrics for classifying malware [2, 4, 5, 7, 10, 11, 15]. However, these methodologies are weak to detect malware variants generated by polymorphism techniques. Their call graph based methods are mainly based on similarity searching [1, 3, 6, 8]. Their adopted similarity matching algorithms are computationally heavy and also can cause false positives and negatives. In some cases, it is hard to finish the analysis by similarity matching at the right time. In addition, these methods using graph properties are limited to depicting the structural information of malware itself.

We propose a novel classification method that examines the topological structure of system call graphs and the node properties of system calls in the call graph. We assume that malicious behavior is determined by system calls, so each system call found in malware is influenced by other system calls found in malware. By extension, the system call set has an influence on the behavior of the program. Therefore, we can detect the malicious behavior of programs by generating the system call graph of the malware and exploring the network properties. We prepare a malicious system call dictionary to classify malware families within each malware class. As we examine various social network properties of system calls found in malware and one-hop neighbors that are adjacent system calls to system calls found in malware, we can classify malware families within each malware class.

The main contributions of this paper are as follows. First, we propose a novel classification method that exploits social network properties such as the degree distribution, degree centrality, and average distance. This work is the first attempt to examine the topological structure of system call graphs and the node properties of malicious system calls in the call graph. The feature set from our model makes the malware detection and classification scalable with less heavy computation than graph matching methods. Furthermore, we found that the degree distribution of system calls found in malware approximately follows a power-law. Second, our method

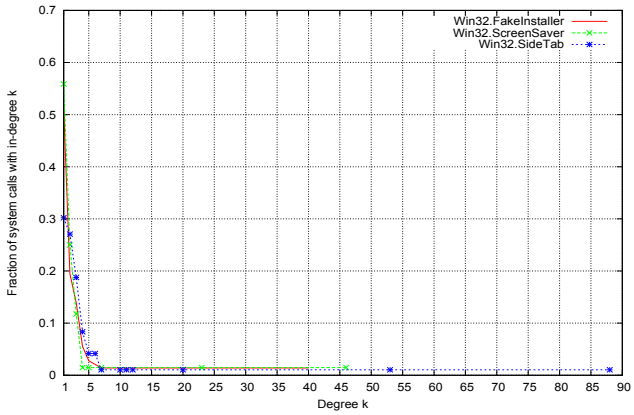

Figure 1: In-degree distribution of malware system calls (adware case)

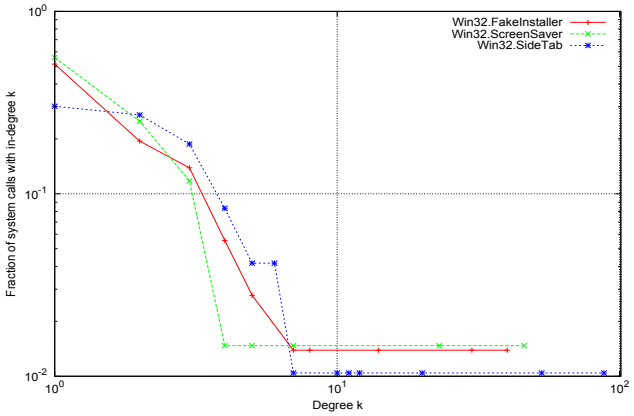

Figure 2: In-degree distribution of malware system calls on logarithmic scales (adware case)

enables AV vendors to react to many species of malicious samples by classifying and matching these with previous ones in a quick and simple way. When building a system call graph, our method requires statistical values related only to social network analysis and imposes no additional computational overhead.

2. DATA EXPLORATION

Based on data exploration, we determine metrics for malware analysis. We used 2,523 malware samples and 123 benign samples in experiments [1]. We build the malicious system call dictionary that is referred to system calls found in malware in previous work [7, 9, 14]. We extract call sequence list of the system calls found in malware and their one-hop neighbors by comparing the call sequences of each sample of malware with the predefined malicious system call dictionary. We transform such call sequence list into a system call graph to examine the social network properties through SNA tool. We disallow the presence of multiple edges but allow the presence of self-edge, when building the system call graph.

We explore the degree distribution of malware system calls. Those system calls with high in-degree centrality are the system calls found in malware related with creating local procedure calls (e.g., *NtAlpcSendWaitReceivePort* and *NtAlpcConnectPort* in adware) and synchronization (e.g., *NtReleaseMutant* in adware). On the other hand, those system calls with in-degree centrality of 1 are mostly composed of system calls in benign related with file creation or opening, section creation, and process creation or termination; a few of system calls found in malware related with process creation (e.g., *NtCreateUserProcess* and *NtCreateWorkerFactory* in adware) and synchronization (e.g., *NtCreateKeyedEvent* in adware) are included. Since system call patterns (e.g., frequency or sequence) determine the program behavior, the malware calls out more system calls found in malware than system calls in benign while conducting its malicious behavior. Thus, if the system calls with higher degree (i.e., the system calls found in malware) are deleted in the system call graph, those networks that perform the malicious behavior can collapse. The in-degree distribution of system calls in Figure 1 shows that the in-degree distribution of the malware system calls is highly right-skewed. The plot on logarithmic scales in Figure 2 shows that the in-degree distribution of the malware system calls follows power-law with noises in the right-hand side. We then carry out goodness of fit test through power-law hypothesis tool [12] which is statistically principled framework for testing the power-law hypothesis to the case of binned empirical data. We estimate the parameters $p(x_{min})$ and α of the power-law model as shown in Table 1. Since p-value is larger than 0.1, the power-law is a plausible statistical hypothesis for our data set.

Table 1: The result of goodness of fit test

Malware ID	$p(x_{min})$	α	p-value
Win32.FakeInstaller_001	.13889	2.21598	.65634
Win32.ScreenSaver_001	.11765	1.86510	.54446
Win32.SideTab_001	.04167	1.82649	.17682

This implies that the system call does not connect other system calls in a random way. Most system calls are connected to 1 system call and a few system calls are connected to numerous system calls. The portion of those calls with degree of 1 is significantly different among malware families as illustrated in Figure 1. The portion of system calls with degree of 1 gives hints to classify the malware families. Networks with power-law degree distribution are called scale-free networks, and a scale-free network provides a number of interesting network properties in terms of the degree distribution, centrality, and cohesion. We use network properties of the system call graphs to detect and classify malware. The portion of system calls with in-degree of 1 is much higher than that of system calls with in-degree over 1, and it varies according to malware families. Thus, the difference in the portion of system calls with in-degree of 1 can be a major metric for classifying malware. The out-degree distribution of malware system calls is similar to the in-degree distribution. We thus choose these portions two portions of the system calls as principal features to classify malware: portion of calls with in-degree of 1 and portion of calls with out-degree of 1. Note that the plot for weighted in- and out-degrees of 1 has approximately the same shape as that for in- and out-degrees of 1. We also explore the average in-degree centrality and average weighted in-degree centrality and average distance of the malware system call graph. As a result, all those metrics are also good features for classifying malware.

3. SYSTEM OVERVIEW

After finding social network properties that distinguish behavioral traits of each malware sample, we implemented an automatic classification system for malware analysis. When a malicious program is passed to our system, our system extracts system call found

[1] Our dataset is available at http://ocslab.hksecurity.net/mal_netminer

Table 3: Discriminatory ability for 3,615 malware and 123 benign samples

Classifier	Accuracy			AUC		
	adware+benign	trojan+benign	worm+benign	adware+benign	trojan+benign	worm+benign
Naive Bayes	**99.99**	92.34	97.72	**1.00**	**0.97**	0.99
Boosted NB	99.96	93.88	99.05	**1.00**	0.95	0.99
RIPPER	99.66	93.70	98.05	0.99	0.89	0.95
Boosted RIPPER	99.71	**95.43**	99.06	**1.00**	**0.97**	**1.00**
RBF	99.94	93.32	94.95	**1.00**	0.94	0.99
Boosted RBF	99.98	94.16	98.69	**1.00**	0.97	**1.00**
C4.5	99.68	94.20	98.48	0.99	0.87	0.97
Boosted C4.5	99.69	95.42	**99.18**	**1.00**	0.96	**1.00**
K-NN	99.88	94.69	98.43	**1.00**	0.95	0.99
Boosted K-NN	99.82	93.23	98.49	0.99	0.95	0.99

Table 2: Malware samples for experiments

Malware Class	Malware Family	Quantity
adware (1,820)	Win32.FakeInstaller	435
	Win32.ScreenSaver	1,211
	Win32.SideTab	174
trojan (742)	Win32.Llac	352
	Win32.Pakes	176
	Win32.Regrun	214
worm (1,053)	Win32.Mydoom	646
	Win32.Mytob	358
	Win32.Zwr	49

in malware and their one-hop neighbors to examine their social network properties. The system then compares the graph properties in the malicious program with those in each malware family, and classifies malware. Our system is composed of three modules: the Behavior Identification Module, the SNA Metric Extraction Module, and the Classification Module. The Behavior Identification Module executes the given malicious program for 30 seconds and captures system call sequences through Nt-Trace. The Behavior Identification Module then compares them with pre-defined malicious system call dictionary, extracts system call found in malware and their one-hop neighbors, and makes the refined system call sequences to depict a system call graph. The SNA Metric Extraction Module extracts the social network properties: portion of calls with in/out-degree of 1, portion of calls with weighted in/out-degree of 1, average in-degree centrality, average weighted in-degree centrality, and average distance. For an automatic classification, we choose naive Bayes, RIPPER, RBF, decision tree (C4.5), K-NN, and boosting (AdaBoost) classifiers. The Classification Module utilized WEKA.

4. PERFORMANCE EVALUATION

4.1 Experiment setup

The 3,615 malware samples listed in Table 2 were collected from December 2012 to March 2013 through web crawling and malware repository such as virusshare [2]; duplicated malware samples were eliminated according to SHA 256. We also excluded malware samples diagnosed by fewer than 20 AV vendors included by the VirusTotal [13] dataset. We used textual description of malware that were produced by Kaspersky Lab [3]. We used 123 benign samples in the Windows system32 directory. We used 5-fold cross-validation to evaluate the performance in our experiments; k-fold cross-validation is to make the whole data set randomly partitioned into k equal size subsamples, a single subsample was used as the validation data for testing the model, and the other k - 1 subsamples were used as training data. We repeated the cross-validation process 5 times.

[2]retrieved from `http://virusshare.com/`

[3]retrieved from `http://www.kaspersky.com/`

Table 4: The importance of feature attributes through information gain attribute ranking method

Feature attribute	benign + adware	benign + trojan	benign + worm
Average in-degree centrality	**0.995**	0.440	0.667
Average weighted in-degree centrality	0.437	0.387	0.755
Portion of calls with in-degree of 1	0.964	**0.614**	0.681
Portion of calls with out-degree of 1	0.817	0.429	**0.789**
Average distance	0.651	0.279	0.691
Portion of calls with weighted in-degree of 1	0.257	0.309	0.610
Portion of calls with weighted out-degree of 1	0.436	0.434	0.763

4.2 Experiment results and analysis

Our performance evaluation focuses on the effectiveness of classification ability of malware. We demonstrate that our system performs well in detecting and classifying malware families within each malware class. We used the accuracy as the performance metric, since the metric for performance evaluation must focus on the predictive capability of the model. The performance of malware classification model is determined by how well the model classifies various pieces of malware. Moreover we used Receiver Operating Characteristic (ROC) curve as the method for comparing classification models. To compare the ROC performance of classifiers intuitively, we calculated the AUC of each classifier, since the AUC represents the ROC performance in a single scalar value. Table 3 shows that the boosted RIPPER, boosted C4.5, and K-NN classifiers outperform the other classifiers in terms of accuracy. The boosted RIPPER classifier slightly outperforms the boosted C4.5 and K-NN classifier. The boosting algorithm somewhat improves the classification performance of all classifiers except for the K-NN classifier. Several factors may have affected the experiment results. Since the social network properties of some trojan samples, such as the average in-degree centrality and portion of system calls with in-/out-degree of 1, are similar to those of benign sam-

ples, classifiers need the discriminatory ability between some trojan and benign samples that are non-linearly separable. The reason is because the RIPPER and C4.5 classifier underperform other classifiers which enable to separate malware non-linearly. In the case of the K-NN classifier, the lack of general information has influence on overall performance. Since the RBF classifier has some limitation to classify data samples with a large Mahalanobis distance from RBF center, the RBF classifier misclassifies some benign samples as trojan samples. In the case of naive Bayes, the assumption of class conditional independence causes loss of accuracy. Through our experiment evaluation, the boosted RIPPER and boosted C4.5 classifier were found to outperform the other classifiers in terms of accuracy and AUC. Despite the boosted RIPPER classifier performed a little better than the boosted C4.5 classifier, the difference of performance was included in performance variation of each classifier. Thus, we conclude that the boosted RIPPER classifier is the best classifier in the perspective of effectiveness of malware classification and detection ability. When a classifier learns training dataset, the classifier decides optimal weight of feature attributes with considering importance of feature attributes. We study what feature attribute has relatively larger effect on detecting and classifying malware. We measure the importance of feature attributes in the manner of information gain attribute ranking in Table 4. Table 4 shows that how much each feature attribute has effect on detecting and classifying malware, when utilizing information gain attribute ranking method. The rank of importance of feature attributes changes according to dataset, but the top three importance of feature attributes is constant. Then, the proposed method can classify the malware with fewer features than previous methods adopting call graph in malware detection and classification. Moreover, our proposed method only takes on average 20 seconds to classify malicious executable; we performed all experiments in Intel(R) Xeon(R) X5660 and 4GB of RAM with 32-bit Windows 7 professional K. The majority of that time is spent time to measure social network properties from Ucinet 6. Since Ucinet 6 supports neither CLI (Command Line Interface) nor relevant API, more reduction of SNA analysis time is out of scope. Nonetheless the metrics that measure the social network properties of system calls found in malware enable us to build a quick and simple detection system against malware.

5. CONCLUSION

In this paper, we have presented a novel malware classification method that exploits social network properties. Our proposed method is the first attempt to examine the topological structure of system call graphs and the node properties of malicious system calls in the call graph. We classified malware by exploiting the social network properties extracted from the system call graphs, which are implicitly equivalent to distinct behavioral characteristics. Our experiments demonstrate that the proposed method performs well in classifying malware families within each class and detecting malware from benign programs. Experiment results show that our method with the boosted RIPPER outperforms the other classifiers. Our method surpassed more than 98% classification accuracy in each malware class.

6. ACKNOWLEDGMENTS

This research was supported by the MSIP(Ministry of Science, ICT & Future Planning), Korea, under the C-ITRC (Convergence Information Technology Research Center) support program (NIPA-2013-H0301-13-3007) supervised by the NIPA (National IT Industry Promotion Agency).

7. REFERENCES

[1] B. Anderson, D. Quist, J. Neil, C. Storlie, and T. Lane. Graph-based malware detection using dynamic analysis. *Journal in Computer Virology*, 7(4):247–258, 2011.

[2] U. Bayer, P. Comparetti, C. Hlauschek, C. Kruegel, and E. Kirda. Scalable, behavior-based malware clustering. In *Proceedings of the 16th Annual Network and Distributed System Security Symposium (NDSS '09)*, 2009.

[3] D. Bruschi, L. Martignoni, and M. Monga. Detecting self-mutating malware using control-flow graph matching. In *Proceedings of the 3rd International Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA '06)*, pages 129–43, 2006.

[4] M. Christodorescu and S. Jha. Static analysis of executables to detect malicious patterns. In *Proceedings of the 12th USENIX Security Symposium (SSYM '03)*, pages 169–86, 2003.

[5] J. Dai, R. Guha, and J. Lee. Efficient virus detection using dynamic instruction sequences. *Journal of Computers*, 4(5):405–14, 2009.

[6] X. Hu, T. Chiueh, and K. Shin. Large-scale malware indexing using function-call graphs. In *Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS '10)*, pages 611–20, 2009.

[7] C. Kolbitsch, P. Comparetti, C. Kruegel, E. Kirda, X. Zhou, and X. Wang. Effective and efficient malware detection at the end host. In *Proceedings of the 18th USENIX Security Symposium (SSYM '09)*, pages 351–66, 2009.

[8] J. Lee, K. Jeong, and H. Lee. Detecting metamorphic malwares using code graphs. In *Proceedings of the 2010 ACM Symposium on Applied Computing (SAC '10)*, pages 1970–7, 2010.

[9] L. Martignoni, R. Paleari, and D. Bruschi. A framework for behavior-based malware analysis in the cloud. In *Proceedings of the 5th International Conference on Information Systems Security (ICISS '09)*, pages 178–92, 2009.

[10] A. Mohaisen and O. Alrawi. Unveiling zeus: Automated classification of malware samples. In *Proceedings of the 22Nd International Conference on World Wide Web Companion*, WWW '13 Companion, pages 829–832, 2013.

[11] S. Tabish, M. Shafiq, and M. Farooq. Malware detection using statistical analysis of byte-level file content. In *Proceedings of the ACM SIGKDD Workshop on CyberSecurity and Intelligence Informatics (CSI-KDD '09)*, pages 23–31, 2009.

[12] Y. Virkar and A. Clauset. Power-law distributions in binned empirical data. *arXiv preprint arXiv:1208.3524*, 2012.

[13] Virustotal. Virustotal, free online virus and malware scan. https://www.virustotal.com/. Accessed Dec. 10, 2013.

[14] X. Wang, Z. Li, N. Li, and J. Choi. PRECIP: Towards practical and retrofittable confidential information protection. In *Proceedings of 15th Network and Distributed System Security Symposium (NDSS)*, 2008.

[15] D. Yuxin, Y. Xuebing, Z. Di, D. Li, and A. Zhanchao. Feature representation and selection in malicious code detection methods based on static system calls. *Computers & Security*, 30(6):514–24, 2011.