Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 46 (2015) 804 – 811

Procedia Computer Science

International Conference on Information and Communication Technologies (ICICT 2014)

Integrated static and dynamic analysis for malware detection

P. V. Shijo[a,*], A. Salim[b]

[a]Department of Computer Science and Engineering, College of Engineering Trivandrum, India
[b]Department of Computer Science and Engineering, College of Engineering Trivandrum, India

Abstract

The number of malware is increasing rapidly regardless of the common use of anti-malware software. Detection of malware continues to be a challenge as attackers device new techniques to evade from the detection methods. Most of the anti-virus software uses signature based detection which is inefficient in the present scenario due to the rapid increase in the number and variants of malware. The signature is a unique identification for a binary file, which is created by analyzing the binary file using static analysis methods. Dynamic analysis uses the behavior and actions while in execution to identify whether the executable is a malware or not. Both methods have its own advantages and disadvantages. This paper proposes an integrated static and dynamic analysis method to analyses and classify an unknown executable file. The method uses machine learning in which known malware and benign programs are used as training data. The feature vector is selected by analyzing the binary code as well as dynamic behavior. The proposed method utilizes the benefits of both static and dynamic analysis thus the efficiency and the classification result are improved. Our experimental results shows an accuracy of 95.8% using static, 97.1% using dynamic and 98.7% using integrated method. Comparing with the standalone dynamic and static methods, our integrated method gives better accuracy.

Peer-review under responsibility of organizing committee of the International Conference on Information and Communication Technologies (ICICT 2014)
Keywords: Malware detection, Malicious software, Static analysis, Dynamic analysis, Malware classification, Machine learning, N-gram, Feature Extraction

1. Introduction

The Internet is becoming an important part of people's everyday life as the online payments and online banking is being popular nowadays. The users of Internet including corporates faces security threats caused by malware. Malware (malicious software) refers to programs that affect a computer system without the user's permission and with an intention to cause damages to the system or steal private information from the system. Depending on the behavior and the way of infection malwares are classified as viruses, worms, Trojan Horses, root-kits, spy-ware etc.

Thousands of new malwares are emerging every day and the existing malwares are evolving in their structure become difficult to detect. In 2012 McAfee Labs identified more than 75 million new malware samples resulting in an average of 55,000 new instances of malware identified per day[17]. As Figure 1 illustrates, this number has increased

* Corresponding author. Tel.: +91-944-737-0635.
E-mail address: shijovijayan@gmail.com

Peer-review under responsibility of organizing committee of the International Conference on Information and Communication Technologies (ICICT 2014)
doi:10.1016/j.procs.2015.02.149

dramatically over the past few years. Similarly, Panda Labs reported more than 60,000 new malware samples per day in 2013 and an average of more than 73,000 per day in the first quarter of 2014[18] .

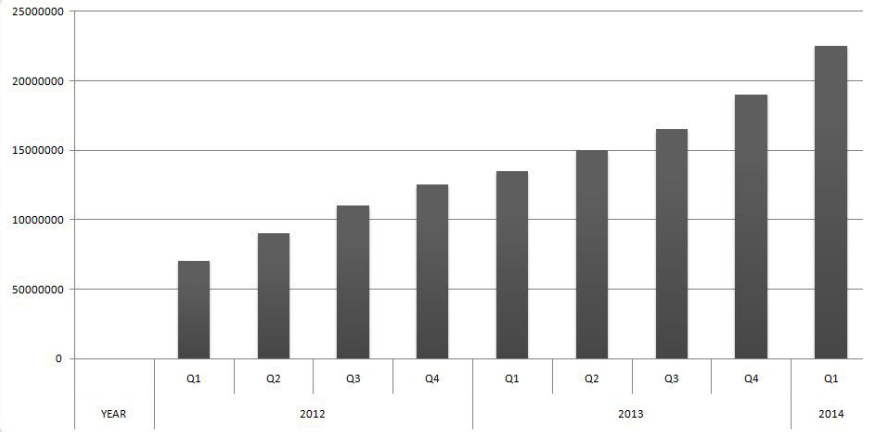

Fig. 1: Total number of malware samples identified by McAfee Labs during 2012 to 2014

Due to this vast amount of new samples emerging every day, security specialists and antivirus vendors depends on automated malware analysis tools and methods in order to distinguish malicious from benign code[4] . Most of the commercial anti-virus software uses signature based malware classification method[1,4,5]. It is a method of identifying unknown malware programs by comparing them to a database of known malware programs. The signature is a unique identification of a binary file. Signature may be created using static, dynamic or hybrid methods and stored in signature databases. Because new malwares are being created each day, the signature based detection approach requires frequent updates of the virus signature database which is the main disadvantage of the method. In static analysis features are extracted from the binary code of programs and are used to create models describing them. The models are used to distinguish between malware and legitimate software. The static analysis fails at different code obfuscation techniques[11] used by the virus coders and also at polymorphic and metamorphic malwares[6]. But there are advantages to static analysis that the binary code contains very useful information about the malicious behavior of a program in the form op-code sequence and functions and its parameters.

On the other hand code obfuscation techniques and polymorphic malwares fails at dynamic analysis[14] because it analyses the runtime behavior of a program by monitoring the program while in execution. The main advantage is that it analyses the runtime behavior of a program which is hard to obfuscate[2,15]. But there are some limitations to dynamic analysis. Each of the malware sample must be executed within a secure environment for a specific time for monitoring the behavior. The monitoring process is time consuming and it must ensure that the execution malware cannot infect the platform[12]. The secure environment is quite different from a real runtime environment and the malware may behave in different in the two environments, causing an inexact behavior log of the malware[5]. In addition some actions of malware are activated or triggered under some certain conditions (system date and time or some particular input by the user) may not be detected by the secure virtual environment[1]. But dynamic analysis is a necessary complement to static approach as it is very much preventive against code obfuscations.

Both static and dynamic methods have their own advantages and disadvantages. So a combined method that utilizes both static and dynamic features will be promising in the malware classification. The proposed method, uses both static and dynamic features of malwares and by using machine learning techniques, provides an efficient automated classification of malwares.

2. Related Works

In this section, we will discuss an overview of some of the works related to static and dynamic methods in malware classification.

Z. Salehi et al. proposed a malware detection based on API calls and their arguments[9]. The method uses API calls and a combination of API calls and their arguments as features and analysed their effect on the classification process. Feature selection algorithms are used to reduce the number of features. The experimental evaluation of the method shows an accuracy of 98.4% in the best case using random forest (RF) algorithm. R. Tian et al. proposed an approach for differentiating malicious files from benign files by analysing the behavioural characteristics using logs of various API calls[12]. According to the frequency of occurrence of an API and a threshold value, they selected a list of APIs. Each of the API in the list is a feature. The feature vector is a binary valued vector with 0 for the absence and 1 for the presence of that API. The experiment was on 1368 malware and 456 benign files and the results shows an accuracy of 97%. Rafiqul Islam et al. presented classification method based on combined static and dynamic analysis[1]. For each executable file function length frequency and printable string information were collected and are represented as vectors and that forms the static features. The dynamic feature comprising API function names and parameters. The three feature vectors are combined to form the integrated feature vector. The evaluation results shows that the classification accuracy of about 97.05%. Younghee Park et al. proposed malware behaviour identification by the use of graph clustering[3]. A Kernel Object Behavioural Graph (KOBG) is constructed from the behaviour information. Given the KOBGs of a set of malware instances belonging to the same family, a common behavioural graph for the family is obtained through graph mining called the Weighted Common Behavioural Graph (WCBG) for the family. This graph is directed, and has edge weights derived from the KOBGs that are combined. Using the resulting WCBG any variant of the malware family may be detected by a matching process. Chatchai Liangboonprakong et al. proposed static N-gram based approach in their work classification of malware families based on N-grams sequential pattern features[7]. The binary executable is disassembled into hexadecimal string and which is used to extract the N-grams. Then sequential pattern extraction is employed to find the frequently occurred sequences which is used to create the feature vector and classification. The method gives an accuracy of 96.6%.

3. Integrated static and dynamic method

Most of the works in malware classification are either using static analysis or using dynamic analysis methods. The proposed method utilizes advantages of both static and dynamic analysis. Static features are extracted from the binary code. The malware executables were collected from the VirusShare[20] community website. Printable strings information (PSI) is extracted from the binary and which is used as static feature. Dynamic analysis is done by using the tool cuckoo[19] sandbox. Dynamic analysis is mainly focused on system call sequences. By combining the features extracted from the binary code and the behavior of the file in execution might be adequate for a better classification result. The proposed method uses machine learning for the automated classification and detection.

3.1. Architecture of the proposed method

An overview of the proposed method is shown in Figure 2. Static and dynamic analysis is performed on the dataset containing both malicious and benign files. Static analysis is done by extracting the PSI features and dynamic analysis is done by extracting API call sequence. The method is explained in the following sections.

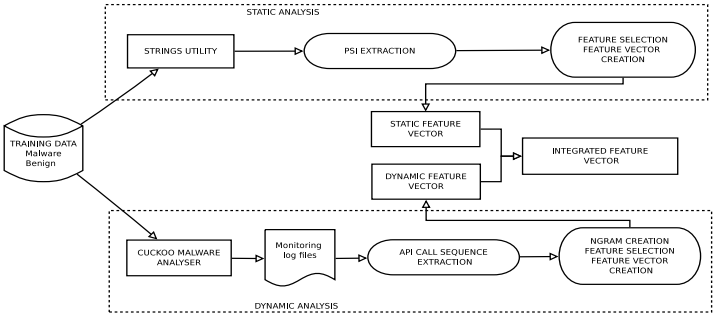

Fig. 2: Architecture of the integrated method

3.2. Static analysis and Static features

Feature extraction process is the major part of any classification task. The static features are extracted from the malware binary files and given as input to various classification algorithms. In this work printable string information (PSI) which is extracted from the binary files is used as the static feature. Printable strings are the un-encoded strings present in the binary executable file. Many literatures shows that PSI is one of the best feature that can be extracted from binary executable [1,13].

Code obfuscation techniques may insert many unwanted PSI to the binary files. So not all the PSI extracted from the binary files are significant and used in the classification. The extracted PSIs are processed so that the output contains strings that are meaningful in the classification. The PSI extracted are sorted according to the frequency of occurrence within a file and PSIs with frequency below a particular threshold are eliminated. A global list of PSI called *feature list* is created which contains all strings that are selected from each of the executable files in the dataset both malware and benign. An entry in the *feature list* is a feature. Each of the malware and benign files are compared with the list and then represented by a binary vector denoting the strings which the malware sample contains or not, recorded as a true/false binary value.

Algorithm 1 shows the process of static feature vector creation. The following example clarifies the static feature extraction and feature selection process. Consider three files corresponding to three binary files after extraction and processing:

```
Data: Dataset :Δ containing malware and benign files f_i
Result: Feature vector and classification results
begin
    foreach f_i ∈ Δ do
        Extract strings from f_i
        Process the raw data to generate useful PSI;
        Create a table of PSI for each f_i sorted according to the frequency;
    end
    foreach f_i ∈ Δ do
        foreach PSI in the table do
            if frequency of PSI > threshold then
                Add PSI to the feature_list;
            end
        end
    end
    Create a binary feature_vector with each PSI in the feature_list as attributes;
    foreach f_i ∈ Δ do
        foreach PSI ∈ feature_list do
            if PSI is present in Table associated with f_i then
                Set value of the attribute in the vector true;
            else
                Set value of the attribute in the vector false;
            end
        end
    end
    Input feature_vector to different learning algorithms in WEKA;
end
```

Algorithm 1: Static feature extraction

File1 : {*GetProcessWindowStation, FindFirstFile, GetLongPathName, HeapReAlloc*}
File2 : {*FindFirstFile, GetLongPathName, GetProcessHeap, GetLastError*}

File3 : {*GetLastError, FindFirstFile, GetLongPathName, GetProcAddress*}

The frequency file is created from these files which will look like as following: Suppose the threshold is set to 2,

Table 1: List of strings extracted from File1, File2 and File3

Printable strings	***Frequency***
FindFirstFile	3
GetLongPathName	3
GetLastError	2
GetProcessWindowStation	1
HeapReAlloc	1
GetProcessHeap	1
GetProcAddress	1

the features selected will be *FindFirstFile*, *GetLongPathName* and *GetLastError*. Then the feature vector for File1 will be as follows:

Table 2: Static feature vector

File	FindFirstFile	GetLongPathName	GetLastError	Class
File1	1	1	0	Benign
File2	1	1	1	Malware
File3	1	1	1	Malware

3.3. Dynamic analysis and Dynamic features

Dynamic analysis is done for extracting the API calls made by a binary file while in execution. In this experiment the cuckoo malware analyser installed under Ubuntu 10.04 with VMware[22] virtual machine is used as the secure environment. Cuckoo is used to run and analyse malware files and generate analysis result of the behaviour of malware while in execution. The log file contains API calls made during execution, registry modifications and the information such as heap memory address and process address.

APIs are provided by the operating system to access the low level hardware through system calls for the application programs. The attackers use the same set of API to do malicious activities. So the presence or absence of an API in the log is not enough to predicting whether the given file is malware or not. In our work we consider the API call sequence. The similarity in the call sequence between files in the same class must be greater than the similarity between the files in the different classes. We use n-gram based method to analyse the call sequence called API-call-grams. As the size of the n-gram increases the number of similar n-grams between two files within the same class itself is very low. On the other hand the analysis based on unigram is same as checking whether the API is present or not in a file. So in our work we consider only 3-API-call-grams and 4-API-call- rams.

The feature vector is created as follows. The set of three and four API-call-grams are generated for each file from the processed call sequence log. For each n-gram set are sorted and grams below a threshold is eliminated. A table for both API-call-grams (3-API-call-grams and 4-API-call-grams) are created in which the entries are API-call-grams from the n-gram set corresponding to a binary file in the dataset. Thus the table contains a global list of API-call-grams which in turn sorted with frequency and we eliminate some API-call-grams with low frequency. The selected API-call-grams constitute the features. Algorithm 2 shows the dynamic feature extraction process. A sample feature vector created by the algorithm is shown in the Table 3.

Data: Dataset :Δ containing malware and benign files f_i
Result: Feature vector and classification results

```
begin
    foreach f_i ∈ Δ do
        - Generate dynamic analysis log file.;
        - Process the log file and extract API call sequence.;
        - Generate 3-API-call-grams and 4-API-call-grams;
        - Sort 3-API-call-grams and 4-API-call-grams with frequency of occurrence;
    end
    foreach f_i ∈ Δ do
        foreach 3-API-call-grams and 4-API-call-grams do
            if frequency of API-call-gram > threshold then
                Add API-call-gram to the corresponding global_list.;
            end
        end
    end
    Sort the global_list of both API-call-grams with frequency of occurrence;
    foreach 3-API-call-grams and 4-API-call-grams do
        if frequency of API-call-gram > threshold then
            Add API-call-gram to the corresponding feature_list.;
        end
    end
    Create a binary feature vector with both 3-API-call-grams and 4-API-call-grams as attribute.;
    foreach f_i ∈ Δ do
        foreach 3-API-call-grams and 4-API-call-grams ∈ feature_list do
            if API-call-gram is present in Table associated with f_i then
                Set value of the attribute in the vector true;
            else
                Set value of the attribute in the vector false;
            end
        end
    end
    Input feature_vector to different learning algorithms in WEKA;
end
```

Algorithm 2: Dynamic feature extraction

class	$3-gram_1$	$3-gram_2$	...	$3-gram_n$	$4-gram_1$	...	$4-gram_m$
Malware	1	1	...	0	0	...	1
Malware	0	0	...	0	1	...	1
Benign	0	1	...	1	0	...	1

Table 3: A sample dynamic feature vector

3.4. The Integrated feature

The proposed method uses the integrated features, which is the feature vector contains both static features and dynamic features. The integrated feature vector is used to classify the binary files. The integrated feature vector will look like as given in Table 4 which a concatenation of both static PSI feature and dynamic API call sequence features.

Class	PSI_1	PSI_2	...	PSI_n	3-GRAMS	4-GRAMS
Malware	1	1	...	0	...	...
Malware	1	0	...	1	...	...
Benign	0	1	...	0	...	...

Table 4: The integrated feature vector

3.5. Machine Learning

Many literature uses the application of machine learning techniques in the malware classification [8,10]. In this work, static and dynamic features are integrated together and the integrated feature vector is used for training and classification. Association vectors, support vector machines, decision tree and random forest are the most used machine learning algorithms for malware classification. But some previous works related to this area shows that support vector machines and random forest are more efficient. Thus our choice will be support vector machines and random forest.

4. Experimental set up and Evaluation

The static analysis is conducted on 997 virus files and 490 clean files each analysed using the strings utility. The experimental environment is set up on an Ubuntu 14.04 machine. In the Ubuntu machine the strings utility is run for each of the binary files. The analysis output for each file is write into a file with the same name as the name of the binary file. From the output file containing PSI, extracted all the strings of length greater than 8 bytes and input to the algorithm to create the feature set. There are 7253 static features extracted in our analysis.

Dynamic feature extraction is done by executing the same binary files used in static analysis in Cuckoo malware analysis system. The malware analyser will produce the log which contains information about the API call sequence. The environment is set up on Ubuntu 10.04LTS operating system. The analyser system is configured to work with virtual machine (VMWare workstation 10.0) inside which we installed three windows XP operating system as the host machines. These machines are called analysis host machines. The binary files are executed in these machines. N-grams are created for API call sequence of each binary file in the dataset and the feature vector is created as explained in the previous section. In our experiment, 5732 4-grams and 3362 3-grams features were selected to create the feature vector. The integrated feature vector is created by concatenating the static and dynamic feature vector together which is used for classification. The WEKA [21] machine learning tool is used for classification.

Table 5 shows the classification results of static, dynamic and integrated methods using SVM and Random forest algorithms.

Method	Random Forest			Support Vector Machine		
	TPR	FPR	Accuracy (%)	TPR	FPR	Accuracy (%)
Static PSI method	0.948	0.150	94.84	0.959	0.078	95.88
Dynamic API-call- grams	0.966	0.100	96.65	0.972	0.099	97.16
Integrated method	0.977	0.063	97.68	0.987	0.026	98.71

Table 5: Classification results of static, dynamic and integrated methods

5. Conclusions

In this work we have presented an integrated approach that uses both static and dynamic features for malware detection. We have proven our thesis that combined static and dynamic features will increase the detection accuracy than stand alone static and dynamic methods.

The results shows that the support vector machine learning technique is best equipped to classify our data. However with random forest also gives better accuracy along with the improvements in the FP and FP rates. From the classification results it is clear that dynamic analysis is better than code based static methods. The dynamic method has more

accuracy than static methods. As with the objective of the study, it is clear that the integrated approach increases the detection accuracy. The integrated method is almost 1.5% better than dynamic analysis with a classification accuracy of 98.7%. Also the results shows that the method has higher accuracy compared with methods in the literature survey.

To continue our work, we will extract more static and dynamic features and reduce the number of features to improve the classification efficiency. Feature selection algorithms can be used to reduce the number of features.

References

1. R. Islam, R. Tian, L. M. Batten, and S. Versteeg. Classification of malware based on integrated static and dynamic features. *Journal of Network and Computer Applications.* vol. 36, pp. 646-656, 2013.
2. M. Ahmadi and A. Sami. Malware detection by behavioral sequential patterns. *Computer fraud and security*, 2013.
3. Y. Park, D. S. Reeves, and M. Stamp. Deriving common malware behavior through graph clustering. in *Computers and security (Elsevier)*, pp. 419-430, 2013.
4. M. Zolkipli and A. Jantan. Malware behavior analysis: Learning and understanding current malware threats. *Second International Conference on Network Applications Protocols and Services (NETAPPS)*. pp. 218-221, Sept 2010.
5. M. Egele, T. Scholte, E. Kirda, and C. Kruegel. A survey on automated dynamic malware-analysis techniques and tools. *ACM Comput. Surv.* vol. 44, pp. 6:1-6:42, Mar. 2008.
6. A. Moser, C. Kruegel, and E. Kirda. Limits of static analysis for malware detection, in *Computer Security Applications Conference, 2007. ACSAC 2007*, pp. 421-430, Dec 2007.
7. C. Liangboonprakong and O. Sornil. Classification of malware families based on n-grams sequential pattern features in *8th IEEE Conference on Industrial Electronics and Applications (ICIEA), 2013*, pp. 777-782, June 2013.
8. Y. H. Choi, B. J. Han, B. C. Bae, H. G. Oh, and K. W. Sohn. Toward extracting malware features for classification using static and dynamic analysis, in *8th International Conference on Computing and Networking Technology (ICCNT)*, pp. 126-129, Aug 2012.
9. Z. Salehi, M. Ghiasi, and A. Sami. A miner for malware detection based on api function calls and their arguments. *16th CSI International Symposium on Artificial Intelligence and Signal Processing (AISP)*, pp. 563-568, May 2012.
10. I. Firdausi, C. Lim, A. Erwin, and A. Nugroho. Analysis of machine learning techniques used in behavior-based malware detection, in *Second International Conference on Advances in Computing, Control and Telecommunication Technologies (ACT)*, pp. 201-203, Dec 2010.
11. I. You and K. Yim. Malware obfuscation techniques: A brief survey. In *International Conference on Broadband, Wireless Computing, Communication and Applications (BWCCA),* pp. 297-300, Nov 2010.
12. R. Tian, M. Islam, L. Batten, and S. Versteeg. Differentiating malware from clean ware using behavioral analysis. in *5th International Conference on Malicious and Unwanted Software (MALWARE)*, pp. 23-30, Oct 2010.
13. M. Islam, R. Tian, L. Batten, and S. Versteeg. Classification of malware based on string and function feature selection in *Cybercrime and Trustworthy Computing Workshop (CTC), 2010* , pp. 9-17, July 2010.
14. H. Zhao, M. Xu, N. Zheng, J. Yao, and Q. Ho. Malicious executables classification based on behavioural factor analysis. In *International Conference on e-Education, e-Business, e-Management, and e-Learning, 2010. IC4E 10*, pp. 502506, Jan 2010.
15. C. Wang, J. Pang, R. Zhao, W. Fu, and X. Liu. Malware detection based on suspicious behaviour identification, in *First International Workshop on Education Technology and Computer Science, 2009. ETCS 09,* vol. 2, pp. 198-202, March 2009.
16. J. R. Crandall, Z. Su, S. F. Wu, and F. T. Chong. On deriving unknown vulnerabilities from zero-day polymorphic and metamorphic worm exploits. In *Proceedings of the 12th ACM Conference on Computer and Communications Security, CCS 05, (New York, NY, USA)*. pp. 235-248, ACM, 2005.
17. McAfee Labs. McAfee threat report: Second quarter 2014. *http://www.mcafee.com/us/resources/reports/rp-quarterly-threat-q2-2014.pdf* 2014.
18. Panda Labs. PandaLabs Threats Report: first Quarter 2014 *http://press.pandasecurity.com/wpcontent/uploads/2014/05/Quaterly-PandaLabs-report_Q1.pdf* . 2014.
19. The Cuckoo sandbox. Accessed 2014. *http://www.cuckoosandbox.org/*
20. VirusShare Malware dataset. Accessed 2014. *http://virusshare.com/*
21. Weka 3: Data Mining open source Software. Accessed 2014. *www.cs.waikato.ac.nz/ml/weka/.*
22. Vmware. Accessed 2014. *www.vmware.com.*