DOI: http://dx.doi.org/10.26483/ijarcs.v8i9.4781

ISSN No. 0976-5697

Volume 8, No. 9, November-December 2017

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

ANDROID MALWARE DETECTION USING HAML

Mahima Choudhary
(M.tech Scholar)
Department of Computer Science Engineering
Apex Institute of engineering and Technology
Jaipur, India

Brij Kishore
(Assistant professor)
Department of Computer Science Engineering
Apex Institute of engineering and Technology
Jaipur, India

Abstrac: The malware is a very common term in today's scenario. It is very harmful for our device. It is continuously gaining the rise in its quantity. It is proving to be a challenging task to detect the malware because whenever we come to evade a technique for its detection, the attackers also evade the new technique to overcome with our detection technique. Presently we have two techniques for the analysis of an application to be a malware or a goodware. these are : static analysis and dynamic analysis Mostly anti-virus software uses signature-based detection technique but it is inefficient in the today's scenario because of the rapid increase in the number and variants of malware. The signature is a unique identifier for a binary file, which is created by analyzing the binary file using static analysis methods. The dynamic analysis uses the actions and behavior during runtime to find out the type of executable (either malware or benign). Both methods have their own benefits as well as drawbacks. This paper proposes a new technique which uses HAML(Hybrid Analysis with Machine Learning).Hybrid analysis is the combined form of static and dynamic analysis to analyses the executable file Machine Learning is used to classify an unknown executable file. In this method, known type of malware and the benign programs are used as training data. By analysis of the binary code and dynamic behavior, the feature vector is selected. The proposed method utilizes the benefits of both static and dynamic analysis thus the efficiency, and the classification result is improved. Our experimental results show an accuracy of 95.87% using static, 97.17% using dynamic and 98.72% using the embedded method. As Compare to the standalone dynamic and static methods, our HAML method gives the more accurate results and is proved to be more efficient.

Keywords: Malware detection; static analysis; dynamic analysis; machine learning

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 face security threats by malicious software. These malicious softwares are known as Malware which is a program that is specially developed to harm the user's device or user's data in a manner such as stealing the private data etc. without giving any notification to the user. Depending on the behavior and the way they infect, malwares are classified as spy-ware, worms, root-kits, viruses, Trojan Horses, etc.

Thousands of new malwares are being developed every day, and the existing malwares are also modifying in their structure,so it becomes very difficult to detect.

Due to the vast amount of new samples emerging every day, security specialists and antivirus vendors depend on automated malware analysis tools and methods in order to distinguish malicious from benign code [1]. Mostly anti-virus products uses signature-based malware classification method [1,2,3]. In this method, malware programs are determined by making comparison of the unknown programs with the known malware programs present in the database. The signature is a antique label provided to a binary file. It can be also called as unique identification. The 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. Static analysis, extracts the features from the binary code of programs and use them to create models. These models are then used to classify the program as a malware or a legal software. The static analysis fails at different code obfuscation techniques[4] used by the virus coders and also at polymorphic and metamorphic malwares[5]. 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.

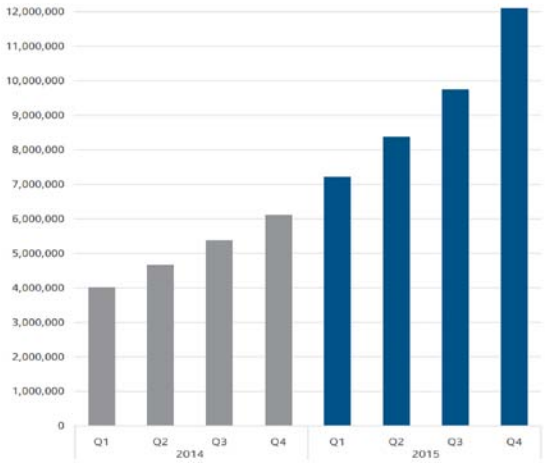

Fig. 1: Total number of malware samples identified by McAfee Labs during 2014 to 2015

On the other hand code obfuscation techniques and polymorphic malwares fails at dynamic analysis[6] 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[7,8]. But there are some limitations to dynamic

analysis. Each of the malware samples 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[9]. 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[3]. 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[2]. 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. PROPOSED ALGORITHM

Embedded static and dynamic method

Mostly the works in malware classification uses either the static analysis or the dynamic analysis methods. But, our proposed method combines the positive aspects of both the methods. We taken the static features from the binary code. Then collected the malware executables from the VirusShare[10] community website. And collected the Printable strings information (PSI) from the binary, which is used as a static feature. The tool cuckoo[11] sandbox is used for performing Dynamic analysis. Dynamic analysis is mainly focused on sequences of the system call. 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.

Architecture of the proposed method

The architecture of the proposed method is shown in Figure 2. The 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.

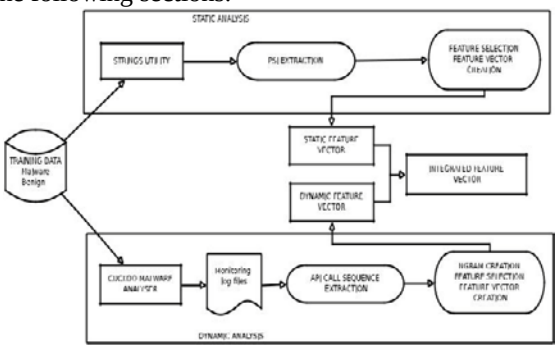

Fig. 2: Architecture of the embedded method

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 show that PSI is one of the best features that can be extracted from binary executable[2,12].

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 a 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 is 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 : containing malware and benign files f_i

    Result: Feature vector and classification results

[9]   begin
 2      for each f_i ∈ do

 3        Extract strings from f_i
 4        Process the raw data to generate useful PSI;
 5        Create a table of PSI for each f_i sorted according to
          the frequency;

 6      end
 7      for each f_i ∈ do

 8        for each PSI in the table do
 9          if frequency of PSI > threshold then

                Add PSI to the feature
10                  list;

11          end

12        end

13      end

14      Create a binary feature vector with each PSI in the
        feature list as attributes;

15      for each f_i ∈ do
16        for each PSI ∈ feature list do
17          if PSI is present in Table associated with f_i then
```

```
                Set value of the attribute in the vector
18              true;
19          else
                Set value of the attribute in the vector
20              false;
21          end
22      end
23  end
24  Input feature vector to different learning algorithms in
    WEKA;
25 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 File

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

Dynamic analysis and Dynamic features

API calls are made by a binary file during its execution. For the extraction of these API calls, Dynamic analysis is performed. The cuckoo malware analyzer is used in this experiment as the secure environment, which is installed under Ubuntu 10.04 with VMware virtual machine. Cuckoo is used to execute the malware files. It analyze the behavior of malware at the of execution and generate the analysis result based on it. 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 predict 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 the n-gram based method to analyze 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 less. 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 shown in the table. The set of 3 and 4 API-call-grams are generated for each file from the call sequence log which is processed. For each file, n-gram set are sorted, and the grams which are below to a threshold are eliminated. A table for both API-call-grams (3-API-call-grams and 4-API-call-grams) is created in which the the data are: the binary file in the dataset and the corresponding API-call-grams from the n-gram set. 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 Table 3.

Data: Dataset : containing malware and benign files f_i

Result: Feature vector and classification results

```
1 begin
2     for each f_i ∈ do
3         - Generate dynamic analysis log file.;
4         - Process the log file and extract API call sequence.;
5         - Generate 3-API-call-grams and 4-API-call-grams;
6         - Sort 3-API-call-grams and 4-API-call-grams with frequency of occurrence;
7     end
```

```
for each f_i ∈ do
    for each 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
    Sort the global list of both API-call-grams with frequency of occurrence;
    for each 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.;
for each f_i ∈ do
    for each 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
    Input feature vector to different learning algorithms in WEKA;
end
```

Algorithm 2: Dynamic feature extraction

Table 3: A sample dynamic feature vector

Class	3-gram$_1$	3-gram$_2$	..	3-gram$_n$	4-gram$_1$	..	4-gram$_n$
Malware	1	1	..	0	0	..	1
Malware	0	0	..	0	1	..	1
Benign	0	1	..	1	0	..	1

The Embedded feature

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

Table 4: The embedded feature vector

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

Machine Learning

Many researchers have already used the machine learning techniques for classifying the malware [13,14].But, in our work, static and dynamic features are combined together and the resultant feature vector fed as input to the machine learning algo for the purpose of training and classification. Association vectors, decision tree, support vector machines, and random forest are the most popular machine learning algorithms, used for malware classification. But literatures shows that random forest and support vector machines are more efficient. Hence we will use random forest and support vector machines.

3. EXPERIMENT AND RESULT

The static analysis is conducted on 997 virus files, and 490 clean files each analyzed 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 of each file is written into a file having the name same as that of the binary file. We extracted all the strings from the output file (containing PSI), which having length greater than 8 bytes and then fed these into the algorithm, as input, to create the feature set. There are 835 static features are extracted in our analysis.

Dynamic feature extraction is done by executing the same binary files used in the static analysis in the Cuckoo malware analysis system. The malware analyzer will provide the log of sequence of API calls. The environment is set up on Ubuntu 10.04 LTS operating system. The analyzer system is configured to work with a 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 on 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, 573 4-grams and 262 3-grams features were selected to create the feature vector. The machine learning tool WEKA[15] is used for classification.

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

Table 5: Results of Classification through static, dynamic and embedded methods

Method	Random Forest			Support Vector Machine		
	TPR	FPR	Accuracy (%)	TPR	FPR	Accuracy (%)
Static PSI method	0.947	0.152	94.83	0.958	0.079	95.87
Dynamic API-call- grams	0.968	0.099	96.66	0.973	0.098	97.17
Embedded method	0.978	0.062	97.69	0.988	0.025	98.72

4. CONCLUSION

In this work, we have presented an embedded 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 achieved show that the support vector machine technique of machine learning 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 embedded approach increases the detection accuracy. The embedded method is found to be 1.5% better than dynamic analysis with a 98.72% classification accuracy. Also, the results show that the method has higher accuracy compared with methods in the literature survey.

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

REFERENCES

[1] 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.

[2] 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.

[3] 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.

[4] I. You and K. Yim. Malware obfuscation techniques: A brief survey. In International Conference on Broadband, Wireless Computing, Com-munication and Applications (BWCCA), pp. 297-300, Nov 2010.

[5] 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.

[6] 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.

[7] M. Ahmadi and A. Sami. Malware detection by behavioral sequential patterns. Computer fraud and security, 2013.

[8] 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.

[9] R. Tian, M. Islam, L. Batten, and S. Versteeg. Differentiating malware from clean ware using behavioral analysis. in 5th International Confer-ence on Malicious and Unwanted Software (MALWARE), pp. 23-30, Oct 2010.

[10] VirusShare Malware dataset. http://virusshare.com/

[11] The Cuckoo sandbox. http://www.cuckoosandbox.org

[12] 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.

[13] 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.

[14] 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.

[15] Weka 3: Data Mining open source Software. Accessed 2014. www.cs.waikato.ac.nz/ml/weka/.