# Proposal of n-gram Based Algorithm for Malware Classification

Abdurrahman Pektaş
The Scientific and Technological Research Council of Turkey
National Research Institute of Electronics and Cryptology
Gebze, Turkey
e-mail:apektas@uekae.tubitak.gov.tr

Mehmet Eriş
The Scientific and Technological Research Council of Turkey
National Research Institute of Electronics and Cryptology
Gebze, Turkey
e-mail:eris@uekae.tubitak.gov.tr

Tankut Acarman
Computer Engineering Dept.
Galatasaray University
Istanbul, Turkey
e-mail:tacarman@gsu.edu.tr

***Abstract*— Obfuscation techniques degrade the n-gram features of binary form of the malware. In this study, methodology to classify malware instances by using n-gram features of its disassembled code is presented. The presented statistical method uses the n-gram features of the malware to classify its instance with respect to their families. n-gram is a fixed size sliding window of byte array, where n is the size of the window. The contribution of the presented method is capability of using only one vector to represent malware subfamily which is called subfamily centroid. Using only one vector for classification simply reduces the dimension of the n-gram space. Experimental results are performed over a fairly large data set, which is being collected through Computer Emergency Response Team (CERT) activities in the National Research Institute of Electronics and Cryptology, to illustrate the effectiveness of the proposed malware classification methodology.**

***Keywords- malware; n-gram based; classification***

## I. INTRODUCTION

The basic definition of malware (malicious software) may be presented as follows: piece of software code that works for the attacker. Malware has great popularity amongst cyber criminals since it offers attractive income opportunities. This popularity makes the malware an important threat for the computing society.

The presented classification approach uses the centroid of the subfamily which is constructed from its samples. Therefore unknown malware classification can be achieved by using low dimension centroid vector which requires less computational work. Experimental study is performed to validate the accuracy of the presented centroid-based approach. Used data set is constituted by the national activities of CERT Coordination Center, which is the national consultation center for computer security incidents [1].

The representation of malware by using n-gram profiles has been presented in the open literature, see for example [2], [3] and [4]. In these studies some promising results towards malware detection are presented. However malware domain has been evolving due to survivability requirements. Malware has to evade anti-virus scanners to perform its functions. Obfuscation techniques have been developed in order to avoid detection by anti-virus scanner. And these techniques disturb n-gram features of binary form of the malware used by the previous work. Similar methodologies have been used in source authorship, information retrieval and natural language processing [5], [6].

The first known use of machine learning in malware detection is presented by the work of Tesauro *et al.* in [7]. This detection algorithm was successfully implemented in IBM's antivirus scanner. They used 3-grams as a feature set and neural networks as a classification model. When the 3-grams parameter is selected, the number of all n-gram features becomes $256^3$, which leads to some spacing complexities. Features are eliminated in three steps: first 3-grams in seen viral boot sectors are sampled, then the features found in legitimate boot sectors are eliminated, and finally features are eliminated such that each viral boot sectors contained at least four features. Size of feature vectors in n-grams based detection models becomes very large so feature elimination is very important in these models. The presented work has been limited by the boot sector viruses' detection because boot sectors are only 512 bytes and performance of technique is degraded significantly for larger size files.

As a historical track, IBM T.J. Watson lab extended boot virus sector study to win32 viruses in 2000 [8]. At this stage, 3 and 4 grams were selected and encrypted data portions within both clean files and viral parts were excluded due to the fact that encryption may lead to random byte sequences. At the first instance, n-grams existed in constant viral parts were selected as features and then, the ones existed in clean files more than a given threshold value were removed from the feature list. In this study, along the use of neural networks boosting was also performed. Results of this study shown that the developed method performance was not sufficient. Schultz *et al.* has used machine learning methods in [9]. Function calls, strings and byte sequence were used as the feature sets. Several machine learning methods such as RIPPER, Naive Bayes and Multi Naive Bayes were applied, the highest accuracy of 97.6% with Multi Naive Bayes was achieved.

Abou-Assaleh *et al.* [3] contributed to the ongoing research while using common n-gram profiles. k nearest neighbor algorithm with k=1 instead of the other learners was used. Feature set was constituted by using the n-grams and the occurrence frequency, where the occurrence frequency is denoted by L. Tests have been done with

 ISBN: 978-1-61208-010-9 

different *n* (ranging from 1 to 10) and *L* (ranging from 20 to 5000) values. Data set used in these experiments was kept fairly conservative of 25 malware and 40 benign files. With this set, test results shown 98% of success. Using the data in [3], the accuracy slightly dropped to the 94% level.

Kolter *et al.* [2] used 4-grams as features and selected top 500 n-grams through information gain measure. They used instance based learners, TFIDF, naive bayes, support vector machines, and decision trees and also boosted last three learners. Boosted decision tree outperformed all others and gave promising results such as ROC curve of 0.996.

While the battle between malware authors and anti-virus producers are continuing, our motivation is to find the statistical method to classify the malware instance by using n-gram features (profiles) of disassembled malware. In our methodology, we use n-gram feature of the malware to classify the malware instance with respect to their family. n-gram is a fixed size sliding window of byte array, where n is the size of the window. For example the "81EDD871" sequence is segmented (represented) into 5-gram as "81EDD", "1EDD8", "EDD87" and "DD871".

This paper is organized as follows: Section 2 proposes the methodology. Section 3 elaborates and computes the accuracy of the proposed methodology. Finally, concluding remarks and future works are presented in Section 4.

## II. SYSTEM DESIGN

As stated in the introduction, current malware samples cannot be analyzed easily based on their statistical features' as in the previous decade because of the increasing use of the obfuscation techniques by the malware authors.

The proposed algorithm consists of preprocessing, training and testing phase. Malware samples are collected through TR-CERT [1] activities in The National Research Institute of Electronics and Cryptology. We classified our dataset by using Microsoft Security Essential (MSE) antivirus tool [17]. In other words, naming of the malware instance is performed by the MSE tool. Malware naming is not a well standardized area where all vendors, players can name and classify malware according to their intentions, and common sense in naming cannot be achieved among the stakeholders [16]. After that preprocessing step, PEid as a useful tool to inspect PE files, is used to dissemble malware instances [18]. We extract a malware instance's n-gram profile through opcode sequences obtained from PEid. We are using opcode sequences instead of byte sequences of the malware.

In our study, machine codes to extract malwares' n-gram profile instead of byte sequences are considered and the n-gram feature space is considerably reduced. In this manner calculations are performed faster and efficiently. Each malware sample is used to determine its subfamily vector which is named as the centroid of the subfamily.

Family of the malware is a descriptor of the malware used to classify malware samples according to their features

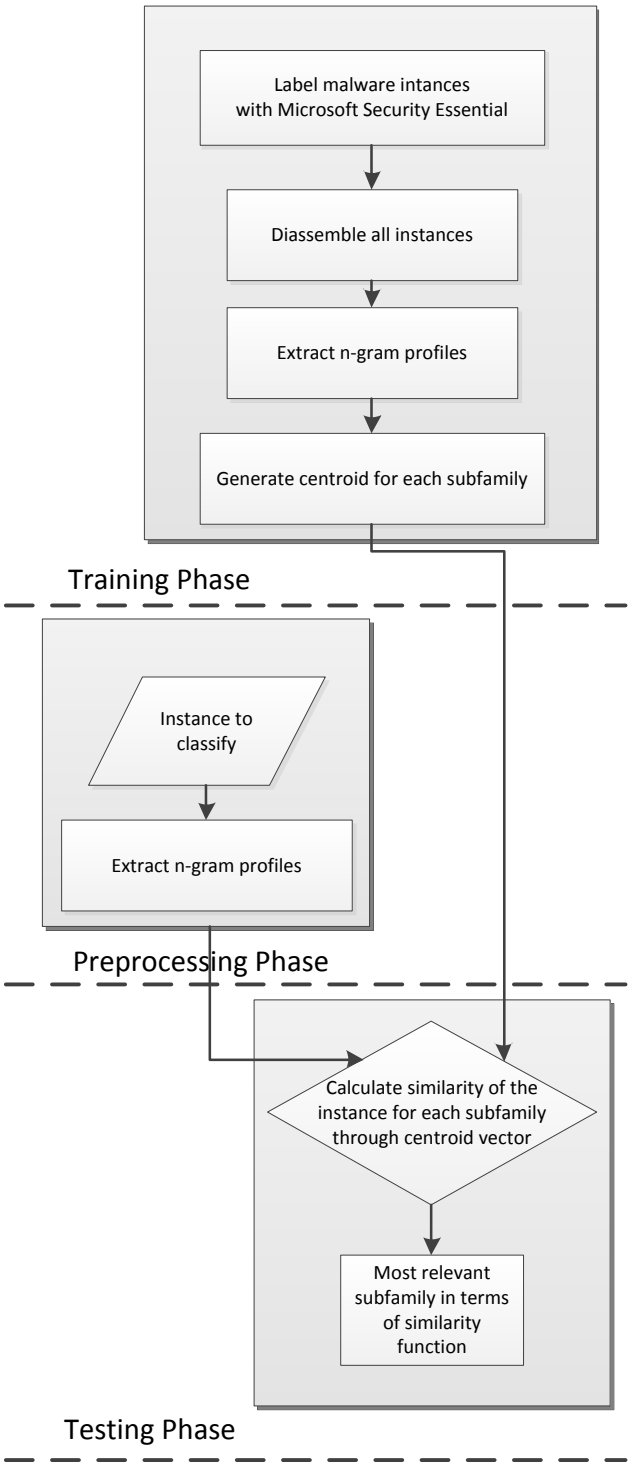


Figure 1. Architecture of the malware classification system

especially in terms of the tasks performed and the purpose of the creation. Subfamily is the specialized version of the family that describes malware samples definitely. For instance if a malware labeled as Win32-Ramnit.F by an anti-virus scanner, this means the malware belongs Win32-Ramnit family and Win32-Ramnit.F subfamily.

 ISBN: 978-1-61208-010-9

Centroid of the subfamily comprises the most frequent n-gram of the subfamily instances. In other words, n-grams (words or terms), which occur with higher document frequency in the subfamily instances, are used to construct the centroid vector. So the subfamily is represented by its centroid vector. For instance, centroid of the subfamily is presented by $\overrightarrow{C_s}$ as follows:

$$\overrightarrow{C_s} = \begin{pmatrix} n-gram\ with\ highest\ df\ value \\ n-gram\ with\ second\ highest\ df\ value \\ \vdots \\ n-gram\ with\ L^{th}\ highest\ df\ value \end{pmatrix}$$

where *df* is the document frequency.

To classify an instance, similarity function is calculated by counting the number of matching n-gram (term) for each centroid of the subfamily.

$$Common\left(C_{s_i}, \overrightarrow{m}\right) = \begin{cases} 1, & (C_{s_i} \epsilon\ \overrightarrow{m}) \\ 0, & otherwise \end{cases} \quad (1)$$

$$Sim(\overrightarrow{C_s}, \overrightarrow{m}) = \sum_{i=0}^{L} Common(C_{s_i}, \overrightarrow{m}) \quad (2)$$

$$Class(\overrightarrow{m}) = \max(\cup_{i=0}^{sub\ number} Sim(\overrightarrow{C_s}, \overrightarrow{m})) \quad (3)$$

where m denotes malware whose family is unknown and it will be determined via presented method. $\overrightarrow{m}$ is the n-gram feature vector extracted from unknown malware instance denoted by m. Subindice is the subfamily indexing for s=1,2…15. The function, denoted by $Common$, returns 1 if malware n-gram profile ($\overrightarrow{m}$) consists *i-th* n-gram of the centroid of taken subfamily($\overrightarrow{C_s}$) denoted by $C_{s_i}$ otherwise return 0. Equation (2) gives similarity measure between the unknown instance and the subfamily centroid. Similarity measure is the sum of the common n-grams. In Equation (3), after all similarity measures are calculated, the unknown instance is classified as the closest centroid's subfamily.

Process flow is illustrated in Figure 1. When an instance has two or more equal similarity value for two different subfamilies, an error occurs. However this error will be named as the small error because these two or more equal similarity values for subfamily may belong to the same family. As we know, the subfamilies sustain their common family feature. Other types of error are named as big error.

## III. EXPERIMENTAL STUDY

In order to perform our experiments, we collect significantly large malware database as stated in the system design section. To obtain more accurate results we count in the subfamilies that contain maximum number of samples in our dataset. In this manner, experiments are carried out 1056 samples belonging to ten families, five of them have two subfamilies, and therefore there exists 15 subfamilies in our dataset. TABLE I indicates how many samples were taken from which subfamily in our dataset. This data set consists only a 2% of the original database. The amount of the sample is sufficient to demonstrate whether n-gram centroid of the subfamily may be used to classify malware instance or may not.

TABLE I. NUMBER OF THE INSTANCES FOR EACH SUBFAMILY

| Subfamily Name | Instance Number | Subfamily Name | Instance Number |
|---|---|---|---|
| Win32-Vobfus.Y | 13 | Win32-Sality.AT | 64 |
| Win32-Alureon.H | 19 | Win32-Small.AHY | 69 |
| Win32-Ramnit.F | 19 | Win32-Renos.NS | 95 |
| Win32-Virut.BG | 19 | Win32-Sality.AM | 100 |
| Win32-Alureon.CT | 22 | Win32-Renos.LT | 137 |
| Win32-Agent.ACF | 23 | Win32-Vobfus.gen!D | 183 |
| Win32-Viking.CR | 30 | Win32-Ramnit.B | 200 |
| Win32-Vobfus.AH | 42 | | |

To evaluate our methodology, five-fold cross-validation is used: the selected malwares' subfamilies are randomly partitioned into five disjoint sets of approximately equal size, named as "folds". Training and testing phases are performed five times. At each iteration step, one fold is selected as a testing set, and other four folds are combined to form a training set. Therefore, each sample is used five times for training and once for testing. And the estimated error is computed as the total error generated from the five iterations, divided by the total number of the initial tuples.

There are two main parameters in the experimental setup: the first parameter is the size of the n-grams and the second parameter is the number of the list size which is constituted by ranking the n-grams according to their df values in the subfamilies. The size of the n-grams, denoted by n, allows us to decide how long in bytes the n-gram will be. In the experiments, tests are run with n=3, n=4, n=5 and n=6. The second parameter, denoted by L, is chosen to express a subfamily in a simple way. Tests are run with L=40, L=50 and L=60.

 ISBN: 978-1-61208-010-9 

TABLE II shows the obtained training error over the parameters n and L as well as TABLE III shows the resulting testing error. As can be seen from the TABLE II and TABLE III, to increase the size of the n-gram does not produce accurate results every time. Because if the parameter n increases, n-grams cannot capture the subfamily features, in contrary the selected n-grams can only represent a feature specific of the sample. However, the opposite case, namely if the n is chosen very small, n-grams can mostly become the common feature of the all subfamilies as well as all samples.

We achieved the highest success rate when n=4 as confirmed by the results in [2] also. Elaborating the parameter choice effects, if the parameter L is increased, the error rate decreases. Since the more common n-gram makes it easy to classify instance appropriately. As maintained in the previous section, the n-gram profiles are captured from the disassembled malware, therefore the space of the n-gram decreases dramatically. For all that, L could not be taken more than 60, due to having very small sized n-gram space (*i.e.*, for Win32-Agent.ACF n-gram feature space is 74)

As a result of the experiment, the most appropriate parameter pair is obtained when n=4 and L=60. The obtained training and testing errors rate for n and L pairs from our experiment are listed in the following TABLE II and TABLE III, respectively.

TABLE II. TRAINING ERROR

| N-gram Length | Top L N-gram in the Subfamily Malwares | | | | | |
|---|---|---|---|---|---|---|
| | L=40 | | L=50 | | L=60 | |
| | Total Error | Without Subfamily Error | Total Error | Without subfamily Error | Total Error | Without subfamily Error |
| n=3 | 0.231 | 0.101 | 0.150 | 0.058 | 0.090 | 0.024 |
| n=4 | 0.143 | 0.056 | 0.106 | 0.021 | 0.053 | 0.014 |
| n=5 | 0.124 | 0.041 | 0.109 | 0.024 | 0.058 | 0.015 |
| n=6 | 0.123 | 0.038 | 0.115 | 0.024 | 0.108 | 0.019 |
| n=7 | 0.151 | 0.031 | 0.115 | 0.031 | 0.098 | 0.019 |
| n=8 | 0.125 | 0.041 | 0.124 | 0.037 | 0.111 | 0.028 |

TABLE III. TESTING ERROR

| N-gram Length | Top L N-gram in the Subfamily Malwares | | | | | |
|---|---|---|---|---|---|---|
| | L=40 | | L=50 | | L=60 | |
| | Total Error | Without Subfamily Error | Total Error | Without subfamily Error | Total Error | Without subfamily Error |
| n=3 | 0.262 | 0.109 | 0.184 | 0.066 | 0.131 | 0.038 |
| n=4 | 0.169 | 0.069 | 0.141 | 0.037 | 0.082 | 0.023 |
| n=5 | 0.150 | 0.056 | 0.128 | 0.038 | 0.082 | 0.026 |
| n=6 | 0.143 | 0.043 | 0.140 | 0.027 | 0.134 | 0.023 |
| n=7 | 0.170 | 0.039 | 0.140 | 0.036 | 0.125 | 0.025 |
| n=8 | 0.139 | 0.042 | 0.148 | 0.040 | 0.138 | 0.034 |

## IV. CONCLUSION

In this paper, a methodology for classifying malware instances from disassembled code by using n-gram feature is presented and it is implemented on a fairly large set. Empirical results demonstrate that the proposed methodology may show acceptable performance in practice. Experimental results show that the classification accuracy for training and testing when n and L are chosen 4 and 60, is achieved at their highest success percentage of %99 and %98, respectively, which seem to be very promising versus the other methodologies.

To improve the accuracy of detection, experiments by using large dataset while using variable length n-gram feature vector of the malware is underway.

 ISBN: 978-1-61208-010-9 

## REFERENCES


[1] Turkey Computer Emergency Response Team, available at site: http://www.bilgiguvenligi.gov.tr/certen/index.php, last access on-line May 31, 2011.

[2] J. Z. Kolter and M. A. Maloof, "Learning to Detect Malicious Executable in the Wild", The Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, pp. 470-478, 2004.

[3] T. Abou-Assaleh, N. Cercone, V. Keselj, and R. Sweidan, "n-gram-based Detection of New Malicious Code", The Proceedings of the 28th Annual International Computer Software and Applications Conference, IEEE Computer Society Washington, DC, USA, pp. 41-42, 2004.

[4] I. Santos, Y. K. Penya, J. Devesa, and P.G. Bringas "n-Grams-Based File Signatures For Malware Detection" The Proceedings of the 11th International Conference on Enterprise Information Systems, Volume AIDSS, pp. 317-320, 2009.

[5] S. Burrows and S. M. M Tahaghoghi, "Source code authorship attribution using n-grams", In Proceedings of the Twelfth Australasian Document Computing Symposium, A. Spink, A. Turpin, and M. Wu, Eds. RMIT University, Melbourne, Australia, pp. 32–39, 2007.

[6] G. Frantzeskou, E. Stamatatos, S. Gritzalis and S. Katsikas, "Effective identification of source code authors using byte-level information", In Proceedings of the Twenty-Eighth International Conference on Software Engineering, L. J. Osterweil, D. Rombach, and M. L. Soffa, Eds. ACM Special Interest Group on Software Engineering, ACM Press, Shanghai, China, pp. 893–896, 2006.

[7] G. J. Tesauro, O. J. Kephart, and B. G. Sorkin,"Neural networks for computer virus recognition" , IEEE EXPERT Magazine, pp. 5-6, 1996.

[8] W. Arnold and G. Tesauro, "Automatically Generated Win32 Heuristic Virus Detection", Virus Bulletin Conference, pp. 51-60, September 2000.

[9] M. G. Schultz, E. Eskin, E. Zadok, and S. J. Stolfo, "Data Mining Methods for Detection of New Malicious Executables", Proceedings of the 2001 IEEE Symposium on Security and Privacy, pp. 38-49, 2001.

[10] Z. Liu, N. Nakaya, and Y. Koui, "The Unknown Computer Viruses Detection Based on Similarity" IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences vol.E92-A no.1 pp.190-196, 2009.

[11] A. Walenstein and A. Lakhotia, "The Software Similarity Problem in Malware Analysis", Dagstuhl Seminar Proceedings 06301 Duplication, Redundancy and Similarity in Software, pp.1-10, 2007.

[12] S. B. Mehdi, A. K. Tanwani, and M. Farroq, "IMAD: In-Execution Malware Analysis and Detection", the Proceedings of the 11th Annual conference on Genetic and Evolutionary Computation, pp. 1553-1560, 2009.

[13] S.S Anju, P. Harmya, N. Jagadeesh, and R. Darsana, "Malware Detection using Assembly Code and Control Flow Graph Optimization", Proceedings of the 1st Amrita ACM-W Celebration on Women in Computing in India, article no: 65, 2010.

[14] J.H. Wang, P. S.Deng, Y.S. Fan, L.J Jaw, and Y.C Liu, "Virus Detection Using Data Mining Techniques", In Proceedings of the IEEE 37th Annual International Conference on Security Technology, pp.71-76, 2003.

[15] M. Siddiqui, M. C. Wang, and J. Lee, "A survey of Data Mining Techniques for Malware Detection using File Features", In proceedings of the 46th Annual Southeast Regional Conference, pp. 509-510, 2008.

[16] M. Bailey, J. Oberheide , J. Andersen ,Z. M. Mao, F. Jahanian, and J. Nazario, "Automated Classification and Analysis of Internet Malware", Proceedings of the 10th international conference on Recent advances in intrusion detection, pp. 178-197, 2007.

[17] Microsoft Security Essential, available at site: http://www.microsoft.com/security_essentials/, last access on-line May 31, 2011.

[18] PEiD tool, available at site: http://www.peid.info/, last access on-line May 31, 2011.


 ISBN: 978-1-61208-010-9 