NLP-based Approaches for Malware Classification from API Sequences

Trung Kien Tran
Department of Computer Science
National Defense of Academy
Yokosuka, Kanagawa,Japan
ed17005@nda.ac.jp

Hiroshi Sato
Department of Computer Science
National Defense of Academy
Yokosuka,Kanagawa,Japan
hsato@nda.ac.jp

***Abstract*— In the field of malware analysis, two basic types, which are static analysis and dynamic analysis, are involved in the process of understanding on how particular malware functions. By using dynamic analysis, malware researchers could collect API call sequences that are very valuable sources of information for identifying malware behavior. The proposed malware classification procedures introduced in this paper use API call sequences as inputs to classifiers. In addition, taking advantage of the development in Natural Language Processing field, we use some methods such as n-gram, doc2vec (or Paragraph vectors), TF-IDF to convert those API sequences to numeric vectors before feeding to the classifiers. Our proposed approaches are divided into 3 different methods to classify malware, that is TF-IDF, Paragraph Vector with Distributed Bag of Words and Paragraph Vector with Distributed Memory. Each of them provides us a very good accuracy.**

Keywords—Malware, API Sequence, Paragraph Vectors, TF-IDF, Natural Language Process

I. INTRODUCTION

The number of new malware has been increasing every day. According to a report on Malware trends in 2017 conducted by G-Data, the trend of increasing 32.9% of new malware in 2016 compare to the previous year is continued in 2017 [1]. Recently, in addition to the detection of malicious malware via heuristic and signature-based methods that struggle to keep up with malware evolution, some new methods based on machine learning have been researched. To analysis malware, machine learning algorithms such as Support Vector Machines (SVM), Random Forests (RF), Naive Bayes (NB) or Neural Networks (NN), etc. are used to address the problem of malicious software detection and classification. Both dynamic analysis and static analysis are used to extract features from malwares, some of the approaches are: (i) Using n-gram analysis to describe a sequence of malware's hex values, opcodes, etc. (ii) API calls and function calls have been widely used to detect and classify malicious software, (iii) Malware is processed as an image: the vector of binary string of zeros and ones extracted from malware file was reshaped into a matrix so that malware file could be viewed as a gray-scale image, then use SVM, K-Nearest Neighbor (KNN) [2] or Convolutional Neural Network (CNN) [3] to classify them.

To achieve better results compared to the existing methods, in this paper, we introduce other effective approaches to classify malware types, that is, combinations of TF-IDF, Paragraph Vector and API call sequences. We considered that API calls are the methods that computer programs (malware, benignwares, etc.) use to communicate with the Operating System (OS). This kind of communication is similar to the way people take advantage of language to interact with the others. Each type of malware might have specific API calls' patterns or unique order of API calls; hence, it might be useful to adopt NLP to classify malware families using API call sequences. Four classification models are explored with the proposed methods which consist of SVM, KNN, Multilayer Perception (MLP) and RF.

II. RELATED WORKS

In this section, we review some malware detection and classification researches, which are related to Natural Language Processing field and API Call as in our proposed methods. The two main approaches commonly applied to malware detection are static analysis and dynamic analysis. Static analysis involves disassembling the application in order to extract features but prone to code obfuscation, while dynamic analysis should always be considered as an analyst's first approach that involves running the application to discovering malware functionality.

Nagano et al. [4] used static analysis combine with natural language processing and machine learning classifiers to distinguish between malwares and benignwares. Information of DLL import, assembly code and hex dump are acquired by static analysis of execution files of malware and benign program before being processed by PV-DBOW model to generate feature vectors. Then, these vectors are fed into SVM and k-nearest neighbor classifiers to analyze and classify malwares. Among all the conducted experiments, the classification based on DLL information acquired the lowest accuracy (about 85.1% and 92.1% corresponding to SVM and k-NN respectively), while the combination of all features got the highest score (99.4% with SVM and 99.3% with k-NN).

Another work is adopting DNA sequence alignment algorithms (Multiple sequence alignment (MSA) and Longest Common Subsequence (LCS)) to get the rigor API patterns from categorized API call sequences. The patterns in combination with the critical API sequences were used to decide whether an examined program is a malware or not. An accuracy of this method was quite high (about 99.8%). This research was introduced by Ki Y et al. [5]

There is another malware research related to API sequence and machine learning. This research is proposed by

978-1-5386-0743-5/17/$31.00 ©2017 IEEE

Chandrasekar Ravi et al. [6], which used 3rd order Markov chain to model the Windows API call sequences. This detection system, which use the dataset of 94 benign and 179 malware executables, achieved an accuracy of 90%.

Nakazato J et al. [7] mentioned the method of using API sequence to detect and classify malware as one of their malware detection process. They performed dynamic analysis to automatically obtain the execution traces of malwares. Then, classify malwares into some clusters using their characteristics of the behavior that are derived from Windows API calls in parallel threads. In this method, the authors use two techniques of NLP, namely n-gram and TF-IDF, to deduce the characteristics of malware samples. This method could classify 90% of 2312 malware samples into at most 20 different clusters.

Although there are still many malware analyses related to API call sequences (such as a method of S. Guo et al.[14] with very high accuracy – above 98%) and NLP (e.g., Nagano et al. [4], Peter Teufl et al.[15], etc.), the methods that take advantage of both API sequences and NLP have not been implemented yet. In the following section, we will introduce three experiments that vectorize API sequences in three ways of NLP before classifying with popular machine learning classifiers.

III. Proposed Method

We classify malwares based on the features extracted from the dynamic analysis process of malwares. The proposed methods, in general, are constructed from four steps: (a) Collect malware's behavior characteristics (API call sequences); (b) Feature Extraction; (c) Feature Vectorization and (d) Classification.

A. Collect malware's behavior characteristics

As mentioned in the previous section, to collect specific features from malwares, there are two main ways called static analysis and dynamic analysis. With the static analysis, useful resources are usually used in malware researches such as API calls' log, sequence of bytes, opcode, etc. This kind of analysis is often simple and fast, but seem to prone to self-encrypting (or packed), obfuscation as well as polymorphic malwares. Although there are some limitations to dynamic analysis such as time consuming, specific environment to log malware's execution, it could deal with the hindrances of static analysis since it analyses the runtime behavior of a program while the program is operating. In our proposed methods, we choose API call sequences as malware's features since their order of calls show how malware behaves with an OS such as File IO, Registry read/write, etc., and could be done with some process hooking library like Detour library, EasyHook, etc. Moreover, other methods[5][6][7] that use API calls as input features also have good results; hence, API call sequences could be good resources to identify malware families.

B. Feature extraction

Although API call sequences are used to demonstrate the behavior of malwares, some kinds of malware inject fake API calls to the API call sequences to cover up temporal information. To reduce the effect of the fake API calls as well as split API call sequences got from the hook programs into different single words, we create a feature extraction algorithm using N-Gram. This idea is inspired by the work of S. Guo et al.[14] for splitting API call sequence into some sub-sequences.

A n-gram is simply an n-character slice of a longer string. The gram of every API call sequence is extracted. By doing sliding window operation in every API call sequence, the series of n-length segment sequence are obtained. Each of the segment sequence is called a gram. Before feeding these API's grams into machine learning classifiers, we need to convert them to the numerical representation of information via vectorization process.

C. Feature Vectorization

In this step, we use three different NLP methods, TF-IDF, Paragraph vectors (PV-DM and PV-DBOW), corresponding to our three proposed approaches to vectorize the extracted grams as well as to reduce the dimensionality of features (in case of using Paragraph vectors).

1) Paragraph vectors

Paragraph vectors proposed by Le et al.[8] is an unsupervised algorithm that learns fixed-length feature representations from variable-length pieces of texts, ranging from phrases or sentences to a large documents, and used to predict words in a document with that inputted vectors. Paragraph vectors learns to correlate labels and words, rather than words with other words as in word2vec [9] [10]. This algorithm has a potential to overcome the weaknesses of existing bag-of-words models.

Two opposite models used to create paragraph vectors were given, the Distributed Memory Model of Paragraph Vectors (PV-DM) and Paragraph Vector with Distributed Bag of Words (PV-DBOW). The distributed memory model is similar to CBOW model of "word2vec". The difference is an ID of a document for paragraph vectors, or the paragraph token – "acts as a memory that remembers what is missing from the current context" [8] – or the topic (label) of the paragraph. This token and numbers of word groups are given to the input layer, learned to predict a word w(t) in the output layer. After being trained, the paragraph vectors can be used as features for the paragraph, and could be fed directly to classifiers such as SVM, k-means, etc.

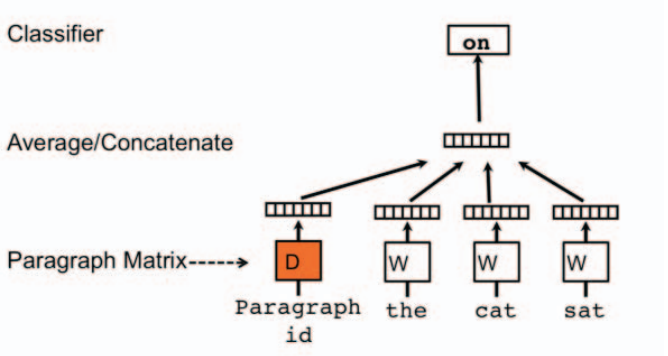

Fig. 1. Paragraph vectors–Distributed Memory model (exceprt from [8])

In contrast, the model of Paragraph vectors without word ordering, the Distributed Bag of Words version of Paragraph

Vector – PV-DBOW, is very analogous to the Word2Vec "skip-gram" mode, but using vector(s) for the text-as-a-whole to predict target words, rather than just vector(s) for nearby-words. First, vectors for ID of a document or paragraph vectors are given to the input layer. Next, the vectors are learned to create vectors in the output layer by predicting words in the document. Finally, vectors of words in a document are outputted after the learning. It is also easy to combine with skip-gram word training.

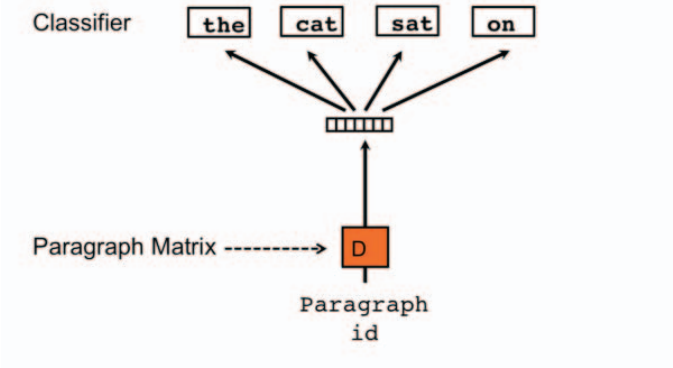

Fig. 2. Paragraph vectors–Distributed Bag of Words model(excerpt from [8])

2) TF-IDF

In short, TF-IDF stands for Term Frequency - Inverse Document Frequency. This measurement is a product of two different measures: Term Frequency (TF) and Inverse Document Frequency (IDF). TF is used to measure the frequency of a term in a document, while IDF shows how much information is provided by that term.

In Nakazato's paper [7], despite using TF-IDF to weight the API sequences, the IDF calculation is quite different from our proposed methods. In his approach, IDF is calculated in regard to the number of appearance of API sequences in each thread and the number of threads. In contrast, we measure IDF of a sequence based on whether that sequence is common or rare across all the malwares in the dataset. It is the logarithmically scaled fraction of a value obtained by dividing the total number of malwares by the number of malwares containing calculating sequence.

D. Classification

After having extracted features for malware classification, machine learning techniques are applied to classify the malware families. In this paper, we use four common classifiers: Support Vector Machine (SVM), K-nearest neighbor (KNN), Random Forest (RF) and Multilayer perceptron (MLP). They all come from the library scikit-learn in Python module. Paragraph vectors which are labels with malware's family name are given for learning and testing.

SVM represents a supervised learning technique suitable for solving classification problems with high dimensional feature space. In this study, we use the SVM with Radial Basic Function (RBF) kernels to train SVM[10]. SVM technique has several methods for not only single-class but also for multi-class classification problem. There are many types of SVM such as linear, non-linear (RBF kernel, Polynomial kernel, etc.), but in general, the purpose of them is to find the most acceptable boundaries between classes.

K-nearest neighbor algorithm is one of the simplest classification algorithm in machine learning. It is also called Instance-based or Memory-based learning. The result from K-NN algorithm depends on the constant k-number. K-NN algorithm uses Euclidean distance to measure the distance between points. The output of this classifier is predicted directly from k-numbers of objects that have the smallest Euclidean distance with the input.

Random forest contains multiple classification trees that are used to obtain results which are combined together for classification. Each of the trees is fed with the same input vector, then outputs that input before the Forest chooses the best result with the most votes.

MLP is a feedforward artificial neural network. Along with Input layers and Output layers, MLP has one or more hidden layers between. Each unit in those layers (except for the input layer) is a neuron that uses a nonlinear activation function. For some specific tasks related to pattern classification, recognition, prediction and approximation, MLP is used for both linear and non-linear problems.

IV. Experiments

A. Dataset

For the experiments, we make use of the dataset implemented and shared by Ki Y [5]. This dataset was built from 23,080 malware samples randomly from Malicia project [12] and VirusTotal [13], and it was shared online (via URL http://ocslab.hksecurity.net/apimds-dataset). The detail of dataset is listed in Table I.

TABLE I. Data description (excerpt from [5])

Category	Subcategory	Ratio (%)
Backdoor		3.37
Worm	Worm	3.32
	Email-Worm	0.55
	Net-Worm	0.79
	P2P-Worm	0.3
Packed		5.57
PUP	Adware	13.63
	Downloader	2.94
	WebToolbar	1.22
Trojan	Trojan (Generic)	29.3
	Trojan-Banker	0.14
	Trojan-Clicker	0.12
	Trojan-Downloader	2.29
	Trojan-Dropper	1.91
	Trojan-FakeAV	18.8
	Trojan-GameThief	0.63
	Trojan-PSW	3.79
	Trojan-Ransomeware	2.58
	Trojan-Spy	3.12
Misc.		5.52

B. Experimental procedure

In this procedure, we implement two different experiments with three ways of classification to compare accuracy. One experiment is to distinguish malwares between two groups:

Trojan and PUP (adware); and the other is a classification of four groups namely, Trojan, Adware (PUP), Worm and the others.

Firstly, we load the dataset and split the API Sequences of every record (malware) into n-gram. Each of record is attached with its own malware family name as a paragraph label (use for paragraph vectors methods). Those split grams are considered as words in a sentence (malware's API call sequence). Secondly, we divide the loaded dataset into four main groups or two groups according to the dataset description. Next, those words are numerically represented by TF-IDF score as well as two models of Paragraph vectors (PV-DM and PV-DBOW). Finally, these converted features are fed into the four different classifiers (SVM, KNN, MLP and RF) to compare an accuracy of methods. For training data, we randomly pick out 80% of total number of records of each malware group. The remain 20% is used for test data. The final accuracy is an average score of 30 times of implementing classification.

C. Results

After some trial and error, we found that 4-gram algorithm performs well, gives the best accuracy as well as reasonable time and resource. Therefore, with this dataset, 4-gram algorithm is chosen to split API sequences. Moreover, with Paragraph Vectors implementation, the parameter window size and vector size are also examined, and with the value of window size 1, vector size 25, we got the best accuracy for two models of Paragraph vectors.

Table II shows the accuracy of three methods using four different classifiers in case of classifying two groups of malware. The highest accuracy, 99.06%, is acquired by SVM using TF-IDF method to vectorize malware features. In case of using KNN (k=3) as a classifier, the model of Distributed bag of words gives us the highest accuracy (98.70%) compared to Distributed Memory and TF-IDF with 97.55% and 98.51%, respectively.

TABLE II. COMPARISON OF 3 METHODS' ACCURACY WITH 2-CLASS CLASSIFICATION

Classifiers	PV-DM	PV-DBOW	TF-IDF
SVM	96.34	98.84	99.06
KNN (k=3)	97.55	98.70	98.51
MLP	96.90	98.11	98.95
RF	96.73	98.69	98.90

Similarly, when classify malware into four groups, the method using TF-IDF still shows the best accuracy among all three proposed methods despite the classifiers. Table III shows the details of these results in case of 4-class classification.

TABLE III. COMPARISON OF 3 METHODS' ACCURACY WITH 4-CLASS CLASSIFICATION

Classifiers	PV-DM	PV-DBOW	TF-IDF
SVM	90.65	93.21	96.16
KNN (k=3)	93.68	95.09	95.25
MLP	92.5	93.97	96.18
RF	93.07	95.12	95.77

Generally, in all conducted experiments above, although the method using TF-IDF outputs the best accuracy among all three methods, the output vectors of TF-IDF have very high dimension, which is related to the number of unique terms in the malware corpus, compared to the others. It could lead to the problem of "the curse of dimensionality" in classifier training. In contrast, using Paragraph Vectors could be more flexible as we can adjust the parameters of those methods to fit the specific dataset such as vector size, window size, etc.

The comparison between our methods and the previous researches, which are used to classify malware groups, is shown in the table IV below. Those methods are all based on static analysis that make use of malware content, and also get high classification accuracy. The difference in the accuracy between our results and the others could be due to the chosen dataset and classification groups.

TABLE IV. COMPARISON WITH OTHER RESEARCHES

References	Features	#samples	#families	Acc.
L.Chatchai et al.[16]	File content	12199	10	96.64
L. Nataraj et al.[18]	File content	63002	531	72.8
R. Tian et al.[17]	File content	1367	11	96.25
Proposed method	API sequence	23080	2	99.06
Proposed method	API sequence	23080	4	96.18

V. CONCLUSIONS

To conclude, NLP becomes more and more powerful, efficient, and has gained much attention from many scientists. Taking advantage of this development could be more beneficial in analyzing malwares. In this paper, the three introduced methods have shown the effectiveness of classifying malwares based on natural language processing. The accuracies of classifications are quite good in all three ways. Furthermore, the accuracy of the classifications could be improved by adjusting the details of parameters of each machine learning classifiers and NLP algorithms. However, due to the lack of sample, we are still not able to evaluate the quality of our methods in case of distinguishing between malware and normal programs.

As future work, we will take a deeper look at NLP algorithms to find more suitable methods for malware analysis. We will also try to collect more benign programs as well as different kinds of malware's API sequences, hence we could re-evaluate our methods to detect and distinguish malwares from benignwares well.

REFERENCES

[1] Ralf Benzmüller, "Malware trend in 2017", 2017. [Online]. Available: https://www.gdatasoftware.com/blog/2017/04/29666-malware-trends-2017.

[2] D. Gibert, J. Bejar, "Convolutional Neural Networks for Malware Classification", Master's thesis. Universitat Politecnica de Catalunya, 2016.

[3] L. Nataraj, S. Karthikeyan, G. Jacob, and B. S. Manjunath, "Malware Images: Visualization and Automatic Classification", Proceedings of the 8th International Symposium on Visualization for Cyber Security, Article No. 4, 2016.

[4] Y. Nagano, R. Uda, “Static analysis with paragraph vector for malware detection”, IMCOM '17 Proceedings of the 11th International Conference on Ubiquitous Information Management and Communication, 2017.

[5] Y. Ki, E . Kim, H . Kang Kim, “A Novel Approach to Detect Malware Based on API Call Sequence Analysis”, International Journal of Distributed Sensor Networks - Special issue on Advanced Big Data Management and Analytics for Ubiquitous Sensors, Volume 2015, January 2015.

[6] Chandrasekar Ravi, R Manoharan, “Malware Detection using Windows Api Sequence and Machine Learning”, International Journal of Computer Applications (0975-8887), Vol 43 (17), 2012, pp. 12-16.

[7] J. Nakazato, J. Song , M. Eto, D. Inoue, K. Nakao, “A Novel Malware Clustering Method Using Frequency of Function Call Traces in Parallel Threads”, IEICE Transactions on Information and Systems Vol. E94.D No. 11, 2011, pp. 2150-2158.

[8] Q. Le, T. Mikolov, “Distributed Representations of Sentences and Documents”, Proceedings of the 31st International Conference on Machine Learning ,China. JMLR: W&CP volume 32, 2014.

[9] T. Mikolov, K. Chen, G. Corrado, J. Dean, “Efficient Estimation of Word Representations in Vector Space”, ICLR Workshop, 2013

[10] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, J. Dean, “Distributed Representations of Words and Phrases and their Compositionality”, Proceedings of the 26th International Conference on Neural Information Processing Systems, 2013, pp. 3111-3119.

[11] Support Vector Machine with scikit-learn. [Online]. Avaiable: http://scikit-learn.org/stable/modules/svm.html

[12] Malicia Project. [Online]. Avaible: http://malicia-project.com/dataset.html

[13] Virus total. [Online]. Available: https://virustotal.com

[14] S. Guo, Q. Yuan, F. Lin, F. Wang, T. Ban, “A malware detection algorithm based on multi-view fusion”, Proceedings of the 17th international conference on Neural information processing: models and applications - Volume Part II,2010, pp. 259-266.

[15] P. Teufl, U. Payer, G. Lackner, “From NLP (Natural Language Processing) to MLP (Machine Language Processing)”, Kotenko, I., Skormin, V. (eds.) MMM-ACNS 2010. LNCS, vol. 6258, 2010, pp. 256–269.

[16] L. Chatchai, S. Ohm, “Classification of malware families based on N-grams sequential pattern features”, Proceedings of the 8th IEEE Conference on Industrial Electronics and Applications (ICIEA '13), 2013, pp. 777–782.

[17] R. Tian, L. Batten, R. Islam, S. Versteeg, “An automated classification system based on the strings of trojan and virus families”, MALWARE: 4th International Conference on Malicious and Unwanted Software, IEEE, New York, N.Y., 2013, pp. 23-30

[18] L. Nataraj, V. Yegneswaran, P. Porras, J. Zhang, “A Comparative Assessment of Malware Classification Using Binary Texture Analysis and Dynamic Analysis”. Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence, 2011, pp. 21-30