# Malware Detection and Classification Based on Extraction of API Sequences

Dolly Uppal, Rakhi Sinha, Vishakha Mehra and Vinesh Jain
Department of Computer Engineering and Information Technology
Government Engineering College
Ajmer, India
dollyuppal2908@gmail.com, srakhi16@yahoo.com, vishakhaashwani@gmail.com, vineshjain1280@gmail.com

***Abstract*— With the substantial growth of IT sector in the 21st century, the need for system security has also become inevitable. While the developments in the IT sector have innumerable advantages but attacks on websites and computer systems are also increasing relatively. One such attack is zero day malware attack which poses a great challenge for the security testers. The malware pen testers can use bypass techniques like Compression, Code obfuscation and Encryption techniques to easily deceive present day Antivirus Scanners. This paper elucidates a novel malware identification approach based on extracting unique aspects of API sequences. The proposed feature selection method based on N grams and odds ratio selection, capture unique and distinct API sequences from the extracted API calls thereby increasing classification accuracy. Next a model is built by the classification algorithms using active machine learning techniques to categorize malicious and benign files.**

***Keywords*— Malware, API sequence, API call gram***

## I. INTRODUCTION

MALWARE is a contraction of two words Malicious Software which cause substantial security threats in computer world. Viruses, Worms, Trojans, Spyware, Adware and other similar attack mechanisms, all fall under the broad term MALWARE. "Malware, in general speaking, are the set of instructions inserted in the application software, with the intention to threaten the security of the system, without the knowledge of its owner." Nowadays, whenever software (application program) runs, it is not sure that it will generate the desired output. It may produce some harmful results or can damage the system. In other words, the system has been attacked by some malicious activity. According to Panda Security Annual Report 2013[1], an average of 82000 new malware has been reported per day. Also, Kaspersky Lab detects 200000 malicious programs every day in 2012[2]. The statistics is large when it comes to system security.

Malware attackers have made windows portable executable (PE) as their most vulnerable prey for carrying out malware based attacks. Therefore, our research work is focused on analyzing portable executables. Malware analysis can be accomplished in two ways- static analysis and dynamic analysis. Static analysis can be defined as the method for examining software without executing it. This technique uses the concept of pattern (byte code-signature) recognition. Anti-Virus Scanners (AVS) follow traditional signature based detection method to identify malware. Signatures are the sequences of bytes existent in the database. Although signature based examination can discern malware by yielding low false positive rates but this technique turns out to be inefficient for malware whose signature are not listed in the database. Also, once should be well versed with the minutiae of software for static analysis, which is not usually possible. Malware detection using static analysis can be easily circumvented by malware writers by making use of the obfuscation techniques. Thus, signature based detection fails to uncover zero day malware attack.

Dynamic Analysis is defined as the method of monitoring the working of an application by analyzing its runtime performance. Day by day Malware are becoming more advanced and vigilant enough so as to halt their execution as soon as they detect that their execution is simulated for security based analysis using emulators or virtual environment. In this way they can easily elude from the malware detection setups. Dynamic analysis can be preferred over static analysis because this technique does not require a deep technical understanding of the software. In addition to this, dynamic analysis is also proficient in detecting malware whose signatures are not known. In present scenario, dynamic analysis is most commonly used to detect malware but it is not adequate because there are some negative aspects of dynamic analysis like some malware exhibit malevolent characteristics of self modification and this type of behavioral aspects cannot be unearthed using this runtime analysis technique. In addition to this, malware which depend on certain trigger conditions for their execution and can alter their behavior when these conditions are achieved, are not exposed by dynamic analysis because all possible execution paths cannot be probed in a single pass.

### *A. Motivation*

Zero day malware attacks by malware assailant are increasing rapidly. Moreover, malware attackers have made windows portable executable as their eminent prey. Code obfuscation techniques, simply frauds antivirus scanners, thereby, decreasing the obfuscated malware detection rate. Therefore, our research work has focused on analyzing portable executables by extracting aspects of API calls.

978-1-4799-3080-7/14/$31.00 ©2014 IEEE

### B. *Offering of the paper*

This paper contributes in following spheres of research:

- A new feature selection method based on N grams and odds ratio selection is used.
- A model is built by the classification algorithms using active machine learning techniques.
- Dataset is collected from other sources, also which are not openly accessible.

The paper is ordered as follows: Section II briefly discusses background and related work. Section III elaborates the proposed work followed by Section IV mentioning the experimental results. Section V concludes our paper.

## II. Background and Related Work

Malware has been a challenge for malware defenders since 1980's. Initially there was no method to defend malware. Gradually, two techniques- signature based detection and anomaly or behavior based detection have been designed for malware detection [3].

Signature based detection uses the technique of pattern recognition in search of predefined signatures stored in a database. Signature based detection technique is commonly used up by AVS as their basic block. This technique does not stand on its hundred percent as it is unable to detect malware whose signature is not present in its database record. On the other hand, behavior based detection mechanism, as its name suggests, keeps an eye on the behavior of suspected application for detection of malicious activity. If something unusual happens, the suspected file is headed as malicious. For detection of unknown malware this detection mechanism stands good.

A thorough study by Robin Sharp [4], illustrated that day by day new variants of malware are being created which has been a tough test for malicious code detectors. New categories of malware- encryption, oligomorphic, polymorphic and metamorphic malwares are taken into account [5]. The first method developed by malware attackers was encryption. Encrypted malware consists of two parts- main body and decryptor. The main body is attacked by the malware, but is no use without its decryptor. These types of malware are easily detectable by antivirus engines. Then comes the oligomorphic malware- they are the advancement of encrypted malware, such that, only the decryptor is made different for each malware, which makes the job of AVS a bit difficult. Polymorphic malware and metamorphic malware are the outcomes of code obfuscation techniques, which are harder to detect, as they easily bypass the AVS.

Study in [6][7] shows that, at present, Portable Executables of Windows are the prime target of malware. The structure of Win32 Portable Executable File Format [8] has been fully illustrated in an article by Matt Pietrek. The analysis of PE can be done through Application Programming Interface (API) calls.

Silvio Cesare and Yang Xiang in their work [9], has given a method to identify polymorphic malware through flow-graphs. The isomorphism of graphs is calculated using previously defined graph isomorphism graph matching algorithms. Using these isomorphic flow-graphs, the similarities among programs are calculated. If the similarity is high between given input program and malware, then the variant of malware is identified. This method encompasses performance rate on real time use like end host email gate-way etc. The proposed approaches by authors succeed in identifying a huge variant of real polymorphic malware.

The authors in [10] proposed a purely dynamic approach to classify benign and malicious samples. The traces of API calls of portable executables are analyzed by executing them on virtual machine. Tracing tools used here are HookMe and Microsoft Detours. The analysis result gives accuracy of above 97%.

In the research of Jain and Nair [11], the emulated environment has been used for extracting API calls. They used QEMU for analysis. The samples are classified using pattern recognition method.

Likewise, authors in [12], first disassembled the portable executables for generating a control flow graph. Control flow graphs are used to generate basic blocks, among which the relation is the show. In order to find the similarity in the executables, the author uses graph isomorphism.

In the study in 2012 [13], the authors worked for developing a new approach against packers. Packing is a technique for covering the malicious code in software through obfuscation technique, which makes the software harmful. To identify the packer tool used for making the software malicious, the authors proposed a technique that generates the signature of every packed malware. The dataset for analysis is divided into two parts. First part is used for constructing the signature for the packer and determining the threshold while second part is created manually through packing each available data set. The first part is the training dataset and second part is the testing data set.

Heejo, Kyoochang and Jeong in their research [14], does not disassembled the executables. They used semantic invariant approach. For differentiating malware and clean files, the author uses the concept of code graph, a topological graph. The code graph is created by using call and jump non-sequential decision making technique, which results in the 67% accuracy of clean files.

Authors in [15], presented a technique for analyzing a malware visually by generating the image metrics out of binary of a malware. This approach classifies the malware according to their families.

A handful of work is done in [16], for detection of malicious code by giving the relation between malware semantics and their API calls. For the complete malware family the author defined a core signature. This signature can define the unknown malware and also the advanced one that lies under the same class of malware family.

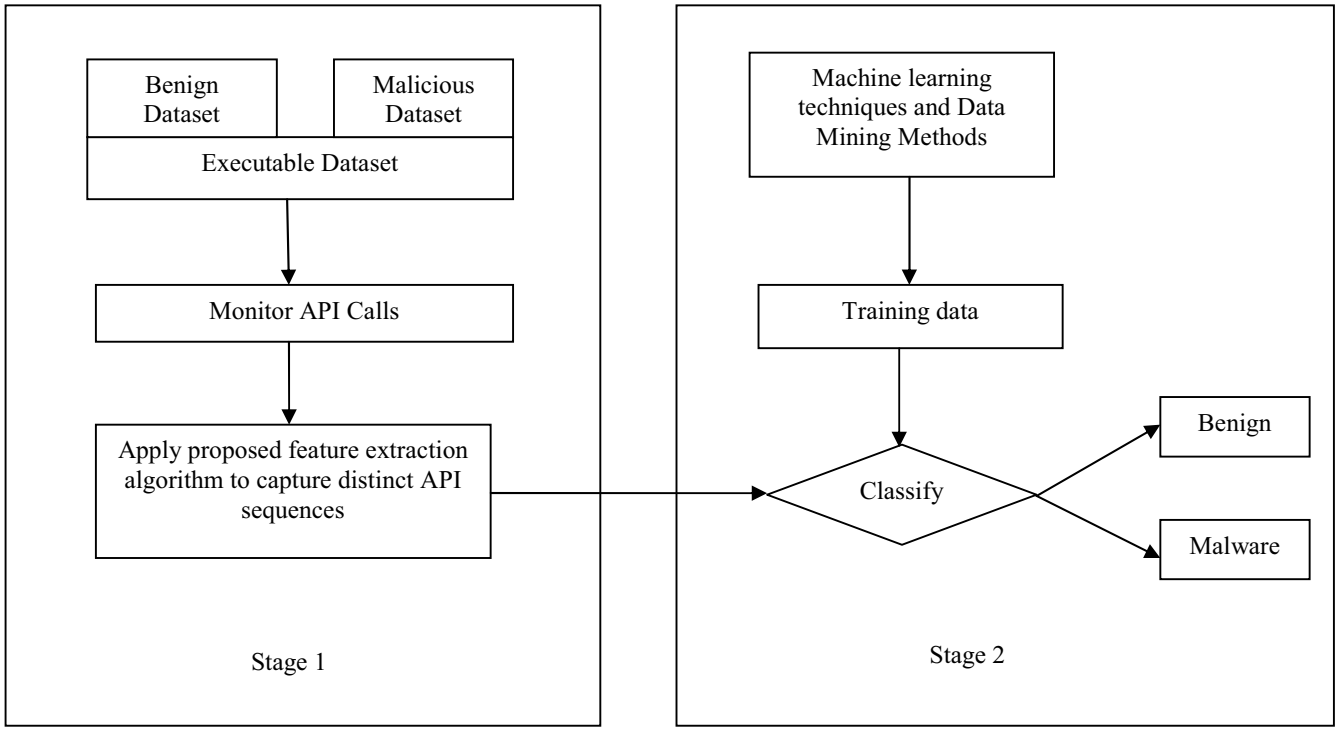


Fig.1 Proposed Malware Identification Framework

The drawback of this approach was only that it does not work for packed malware.

In this research we proposed a novel approach to determine maliciousness in an input sample. First, a pre-modeled program is used to keep track of the execution of the input sample to capture the API calls. Then the next step is to extract the API sequences from these API calls. We proposed an algorithm based on call grams and odds ratio selection to extract the distinct API sequences which are then fed as input to the classifiers to effectively categorize the malicious and benign samples. Classification model is built by making use of machine learning algorithms: Naïve Bayes, Random Forest, Decision Tree, Support Vector machine (SVM).

## III. Proposed Work

This section discusses the proposed methodology comprising of two stages as delineated in Fig.1. Stage1 comprises of two steps; capture the API calls from benign and malicious samples and apply the feature selection technique to API sequences to get distinct API sequences. Stage2 describes the use of machine learning algorithms and classification algorithms to determine whether an input sample is malicious or benign.

### *A. Monitor API Calls*

A pre-modeled program [17] is used to keep track of the execution of input samples and capture the API calls from these samples to get the API call sequences.

### *B. Select the distinct API sequences*

This step is accomplished in two parts; generate the call grams and calculate the odds ratio of each gram and generate feature vector.

The API sequences received in the above step cannot be used for classification purpose. So we proposed feature selection algorithm based on call grams and odds ratio selection algorithm.

*1) Generate the call grams:* First, we generate the calls-grams from the API calls retrieved in step A. 1 API-gram is generated by using API call sequences obtained in step A and by repeatedly performing sliding window operation we can get two, three and up to six API-grams.. Our experimental analysis shows that 2 API-gram give better accuracy than 1API-gram and further improvement in this accuracy is provided by 3 API-gram. Overall, 4 API-gram outperform among all the API-grams.

*2) Calculate the Odds ratio of each call gram and generate feature vector:* Next step is to select distinct and useful call-grams as we retrieved thousands of call-grams in above step. Every obtained API Call-gram is considered as a feature $X_j$ and odds ratio for a feature X can be calculated as

$$OR(X) = \log \frac{\Pr(X \mid c)(1-\Pr(X \mid \bar{c}))}{\Pr(X \mid \bar{c})(1-\Pr(X \mid c))}$$

Where

Pr(X|c) is the probability of feature X appearing in class c.

$\Pr(X \mid \bar{c})$ is the probability of feature X appearing in $\bar{c}$.

c and $\bar{c}$ represent two classes, i.e. benign and malicious.

Next step is to rank the features according to the calculated odds ratio and select the leading n features to form feature vector.

### C. *Machine Learning Algorithms for classification*

Various machine learning algorithms were used to construct the proposed model for classification, some of them being Naïve Bayes (the concept of Bayes conditional probability is used in this classification scheme), Random Forest (multiple trees are used to obtain results which are combined together for classification), Decision Tree (training data is used to generate decision trees according to attributes) and SVM (the nominal attributes are converted to binary ones using this algorithm).

Distinct API sequences were drawn out using proposed feature selection algorithm and were fed as input to all these classification algorithms. These classifiers necessitate the use of supervised learning in order to train them, so we have used K-fold cross validation with all these classification algorithms for examining and producing independent dataset. In K-fold cross validation, thorough training and testing was performed for different values of K and it can be deduced from the findings that error estimation is best in case of K=10 that is, for 10 folds(implies that out of the total data, one tenth data is utilized for testing and nine-tenth data is utilized for training). The performance of all the applied classification algorithms after applying K-fold cross validation method is shown in Fig 2 and the essential inference is that SVM provides best accuracy among all the remaining classifiers.

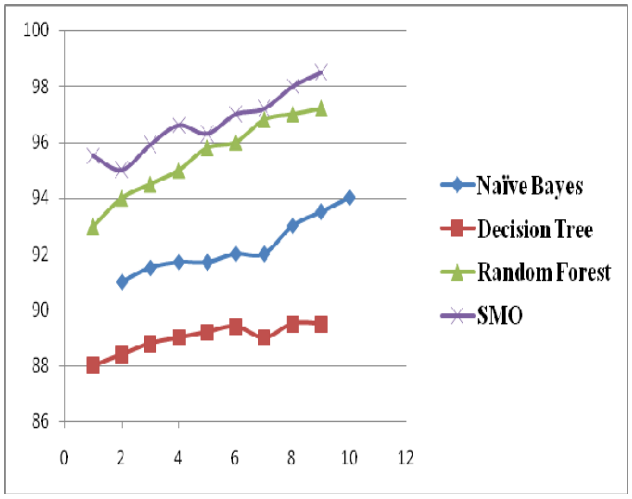


Fig.2 Performance of four different algorithms using k cross validation (k=2 to 10)

## IV. EXPERIMENTAL RESULTS

### A. *Dataset*

Experimental investigation was conducted on the 120 malicious PE files and 150 benign samples. Benign dataset includes various Application softwares like Decision making software, Programming language software, Internet Browser etc. Malicious dataset (refer TABLE II) was collected from [18] including Trojans, Viruses, Rootkits, and Worms.

TABLE I. Benign Dataset

| **Content of Benign samples** | |
|---|---|
| ***Program Category*** | ***Number of samples*** |
| Utility-software | 40 |
| Application software | 50 |
| System software | 35 |
| Device Drivers | 25 |

TABLE II. Malicious Dataset

| **Content of Malicious samples** | |
|---|---|
| ***Program Category*** | ***Number of samples*** |
| Trojan | 35 |
| Worm | 30 |
| Virus | 30 |
| Rootkit | 25 |

### B. *Evaluation Metrics*

Evaluation Metrics can also be constructed by finding TP (True Positive), TN (True Negative), FP (False Positive) and FN (False Negative) where True Positives is all the malware samples recognized correctly as malware, True Negatives is all the benign samples recognized correctly as benign, False Positives is all the benign samples recognized incorrectly as malware and False Negatives is all the malware samples recognized incorrectly as benign.

Accuracy (Acc) is the extent to which malware and benign samples are recognized correctly and can be calculated as

Acc = (TP + TN) / (TP + TN + FP + FN)

Accuracy of discussed algorithms is shown in Fig.3

### C. *Analysis Results*

In this experimental research, designed program captures the API calls from the input sample to get the API sequences. Next, the proposed feature selection algorithm extracts the distinct set of API sequences. Then, these API sequences fed as input to classification algorithms to categorize malicious samples from benign samples. Our experimental analysis inferred that SVM output the best results among all classification algorithms.

A comparison of the accuracy of proposed method to other malware detection methods is listed in Table III.

TABLE III. Comparison of Various Malware Detection Methods

| Authors | Approach | Aspects | ACC, DR, TPR |
|---|---|---|---|
| Ammar Ahmed E. Elhadi et al.[19] | Combine static and dynamic analysis approaches | Combine static and dynamic analysis with behavior and static based methods | Not specified |
| Mojtaba Eskandari et al.[20] | Control Flow Graph based approach | Highlight the problem related to large no. of edges in CFG or large no. of aspects in vector space model | ACC-97.77% DR-97.53% |
| Kyoochang Jeong et al.[14] | Static Analysis | Detection of malwares using code graphs | 67% |
| Faraz Ahmed et al.[21] | Dynamic Analysis | Uses spatial and temporal aspects of API calls | ACC-97% |
| Ronghua Tian et al.[10] | Behavioral Analysis | Examine the behavioral aspects of API calls to differentiate malicious and benign files | ACC-97% |
| Fatemeh Karbalaie et al.[22] | Graph mining techniques | Behavior graph for malware detection | DR-96.6% FP-3.4% |
| Yoshiro Fukushima et al.[23] | Behavioral Analysis | Behavior based identification of suspicious process | DR-60% FP-0% |
| Proposed method | Behavioral Analysis | Malware detection based on extraction of distinct set of API sequences | ACC-98.5% |

## V. CONCLUSION

In this research work, we proposed a novel approach for malware detection method with increased accuracy. We proposed an algorithm based on call grams and odds ratio selection to extract the distinct API sequences which are then fed as input to the classifiers to effectively categorize the malicious and benign samples. After analyzing the obtained results it can be concluded that SVM classification algorithm yields the best results among remaining classifiers.

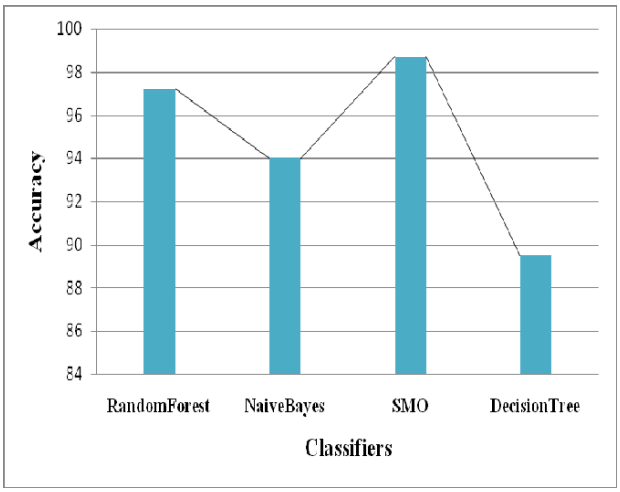


Fig.3 Bar graph showing highest achieved classification accuracy of SMO in comparison to all the discussed algorithms

## REFERENCES

[1] Panda Labs-pandasecurity. www.pandalabs.pandasecurity.com/ .

[2] Kaspersky Security Lab Bulletin. www.kaspersky.in/ 2012.

[3] Vinod P. *et al*., "Survey on Malware Detection",*Workshop on Comput. and Internet Security*,2009,pp.74-79.

[4] Robin sharp, "An Introduction to Malware",Spring 2007.

[5] B.B.Rad *et al*., "Camouflage in Malware: from Encryption to Metamorphism",*Int. J. of Comput. Science and Network Security*, Volume 12, no. 8, August 2012.

[6] Mamoun Alazab *et al.,* "Towards Understanding Malware Behaviour by the Extraction of API Calls", *Second Cybercrime and Trustworthy Computing Workshop*, 2010,pp.52-59.

[7] Parvez Faruki *et al*., "Mining Control Flow Graphs as API Call-Grams to Detect Portable Executable Malware", *In Proc. of the 5th International Conference on Security of Information and Networks* , 2012, pp.130-137.

[8] Matt Pietrek," An In-Depth Look into the Win32 Portable Executable File Format." March 2002, issue of MSDN magazine.

[9] S.Cesare and Y.Xiang, "A Fast Flowgraph Based Classification System for Packed and Polymorphic Malware on the Endhost", *In Proc. of the 24th IEEE Int. Conf. on Advanced Information Networking and Application* ,IEEE,2010,pp.721-728.

[10] Ronghua Tian et al.," Differentiating Malware from Cleanware Using Behavioural Analysis", *In Proc. of the 3rd Int. Conf. on Security of Inform. and Networks*, SIN'10, IEEE, March 2010, pp. 23-30.

[11] V. P. Nair *et al*.,"MEDUSA: MEtamorphic Malware Dynamic analysis Using Signature from API", in *5th Int. Conf.* on *malicious and unwanted software*,ACM, 2010,pp.263-269.

[12] R.T. Dullien and B. Universitaet, "Graph Based Comparison of Executable Objects", *University of Technology*, Florida, 2005.

[13] Smita Naval *et al.,* "SPADE: Signature Based Packer Detection", *In Proc. of the 1st Int. Conf. on Security of Internet of Things*, 2012, pp.96-101.

[14] Kyoochang *et al.*, "Code Graph for Malware Detection", *In Int. Conf. on Inform. Netorking*, ICOIN'08, 2008, pp 1-5.

[15] KyoungSoo Han *et al*., "Malware Analysis Method using Visualization of Binary Files", *In Proc. of the 2013 Research in Adaptive and Convergent Systems*, ACM, 2013,pp.317-321.

[16] V. Sathyanarayan *et al*., "Signature generation and detection of malware families", *In Proc. Of the 13th Australasian Conf. on Information Security and privacy*, 2008,pp.336-349.

[17] APIMonitorCorp.Win32APIMonitor[OL].2011.http://www.APImonitor.com/

[18] VX-Heavens. Virus Collections (VXheavens) http://vl.netlux.org/vl.php/

[19] Ammar Ahmed E. Elhadi et al.," Malware Detection Based on Hybrid Signature Behaviour Applicat. Programming Interface Call Graph", *American J. of Appl. Sci.*, 2012, pp. 283-288.

[20] Mojtaba Eskandari and Sattar Hashemi, "Metamorphic Malware Detection using Control Flow Graph Mining", *Int. J. of Comput. Sci. and Network Security*, IJCSNS, vol. 11,12, Dec. 2011.

[21] Faraz Ahmed, *et al.*," Using Spatio-Temporal Information in API Calls with Machine Learning Algorithms for Malware Detection", *In Proc. of the 2nd ACM workshop on Security and artificial intelligence*, ACM, March 2009, pp. 55-62.

[22] Fatemeh Karbalaie *et al.*, "Semantic Malware Detection by Deploying Graph Mining", *In Proc. of the Student Int. Conf. for IT Security for the next generation*, University of HongKong and Kasperky Academy,2010, pp. 1020-1025.

[23] Yoshiro Fukushima *et al.*," A Behavior Based Malware Detection Scheme for Avoiding False Positive", *In Proc. of 6th IEEE ICNP Workshop on Secure Network Protocols,* (NPSec), Oct., 2010, pp.79-84.