Dynamic Signature-based Malware Detection Technique Based on API Call Tracing

Oleg Savenko[0000-0002-4104-745X], Andrii Nicheporuk[0000-0002-7230-9475], Ivan Hurman and Sergii Lysenko[0000-0001-7243-8747]

Khmelnitsky National University, Khmelnitsky, Ukraine
{savenko_oleg_st@ukr.net, andrey.nicheporuk@gmail.com, devastator192@gmail.com, sirogyk@ukr.net}

Abstract. The paper presents a method for a malware's signature generation based on API call tracing. Technique allows malware detection using a proposed form of signature. The main idea of proposed signature generation is a difference between frequency and interaction of a critical API calls performed by malicious program and benign applications in the process of their own execution. Accordingly the program's behavior signature based on API call tracing consists of two components: the call frequency and the nature of the interaction of critical API calls. An analysis of the first component allows determining the distribution of the critical API calls by groups concerning theirs malicious activity and displays the quantitative component of the signature. An analysis of the second component of the signature provides an opportunity to distinguish malware from benign applications not only in the presence of critical API calls, but also in their interaction with each other. The experimental results showed that the effectiveness of the malware detection using proposed signatures is up to 96.56%.

Keywords: Malware, Cybersecurity, Signature, Behavior, API, API Call Tracing, Chi-Squared Test

1 Introduction

Today, the importance of the problem of cybersecurity is beyond doubt. Since new instance of malware are created and spread faster than the tools are able to identify them, there is always a gap in detection, which leads to computer systems' infection.

According to the McAfee threat report, the number of new samples of malicious software at the end of 2018 has exceeded 60 million [1]. Moreover, the total amount of malware continues to grow in exponentially. This is due to the creation of new technologies and tools for the malware development and improvement of the antivirus evasion techniques [2-4]. Therefore, the development of new methods for malicious software detecting remains an important task.

In this work we propose dynamic signature-based malware detection technique which involves executing a possibly malicious piece of code or executable and detecting its effect upon execution by using specialized monitoring mechanisms. As a feature of detection the API calls that made by executable was chosen. Application Programming Interface or API is a medium communicating layer between Windows

environment and executable. Experimental studies show that malicious and benign programs can be distinguished by API calls frequency performed by them [5].

So our goal is to develop a signature of executable that based on API calls, the analysis of which would allow to separate malware and benign applications.

2 Related works

Today, a number of methods and techniques are used to detect the malware by antivirus tools. The most important is signature analysis. A classic signature-based analysis is based on comparison of a byte patterns or a checksum demonstrates its inefficiency for malicious software that modifies its own code. Therefore, the attention of researchers is focused on the development of new approaches for the signatures generation on the basis of other features that would be able to describe the malware's behavior fully.

In [6] authors are focused on the problem of attack for sensible data with the aid of the virus scanner itself with the use of extracted signatures. The method for automatically deriving signatures from anti-virus software was proposed. Method involves steps: the determination of relevant bytes in each malware sample by utilizing feedback from the virus scanner over multiple runs; aligning the relevant bytes from samples with the same signature and merge them into a single sequence by employing the Needleman-Wunsch algorithm; the transformation the merged sequences into a valid signature format. However, in case then several signatures present in single malware binary, proposed method is not able to recognize any of them.

Authors of DeepSign [7] apply deep belief network (DBN) to solving the problem of malware signature generation and classification. It uses the Cuckoo sandbox to record the execution behavior of each malware. Then, it treats the behavior report as a raw text file and uses uni-grams to convert each report into a 20,000 bit vector. The bit vectors are then fed into deep belief network to generate signatures. Finally, the signatures are fed into a support vector machine for classification. Experiments on 1800 malware samples without benign applications show that DeepSign is able to reach 96.4% accuracy. However, in order to obtain the high reliability of the experiment, the test sets should contain, in addition to the malware, also benign applications, since the rate of false positives is no less important than the accuracy.

In work [8] authors have proposed a method which combines use signature-based and anomaly-based detections. The proposed framework mainly consists of three modules: a database of malware and the PE file, modules of static and dynamic analysis, and a module of classification by similarity analysis. The static analysis consists of the de-obfuscation of packed malware in order to know the packers names and the contribution level of each of them. A dynamic analysis module consists of a virtual environment set up by Cuckoo Sand-Box to run the executable files of malware without infecting the rest of the system. As a result a list of API call sequences that reflect the malware behavior of its code have been used to detect behavior such as network traffic, modifying a file, writing to stderr or stdout, modifying a registry value, creating a process. For classify malware behaviors similarity analysis and various machine learning algorithms were used.

Another API call signature-based approach to malware detection is presented in [9]. To have a higher level of abstraction, related Win-APIs have been mapped to 26

categories, which differ in the nature of the actions performed (files, system registry, etc.), so the behavior of each malware is captured through sequence of these 26 categories of APIs. In order to generate signature Context Triggered Piecewise Hash (CTPH) was computed. The concept of fuzzy hashing has been used as it has the capability to compare two different samples and determine the level of similarity between them. Instead of generating a single hash for a file, piecewise hashing generates many hashes for a file based on different sections of the file.

In [10] a new information technology for botnets detection based on the analysis of the botnets' behaviour in the corporate area network is proposed. Botnets detection is performing combining two ways: using network-level and host-level analysis. One approach makes it possible to analyze the behavior of the software in the host, which may indicate the possible presence of bot directly in the host and identify malicious software, and another one involves monitoring and analyzing the DNS-traffic, which allows making conclusion about network hosts' infections with bot of the botnet. Based on this information technology an effective botnets detection tool BotGRABBER was constructed. It is able to detect bots, that use such evasion techniques as cycling of IP mapping, "domain flux", "fast flux", DNS-tunneling.

The mentioned above methods of detecting viruses have shown a high level of effectiveness, but inserting and executing dummy and redundant API calls can lead to an increase the false positives rate.

Therefore, this study is focused on a problem of creation malware detection method, the basis of signature generation, which is invariant to small scale changes.

3 Dynamic signature-based malware detection technique based on API call tracing

The usage the obfuscation and anti-evasion techniques in malware disables the possibility to isolate a constant part of program code, the analysis of which would make it possible to detect a possible infection. However, it becomes possible using the API calls as the basis for signature, that is, the set of classes, procedures, functions, structures and constants provided by the application or the operating system for use by the external software products.

In order to implement a malware detection process, a new technique has been developed. It involves the following steps:

1. Data preprocessing:
   - 1.1 Signature generation for malware class based on API call tracing of each malware instance;
   - 1.2 Determination of the membership degree of each sample to malware class.
   - 1.3 Construction of a database for malware's behaviors classes and its membership degrees to each classes.
2. Detection of a malicious program represented by signature of program behavior based on API call tracing:
   - 2.1 Monitoring of the executables and their API call tracing;
   - 2.2 Signature construction of the suspicious program;
   - 2.3 The search of the virus signature within the class and the determination whether the suspicious program belongs to one of the malware's class;

2.4 Assignment the malware families (variant).

Let us take a closer look at each step of the method.

3.1 Data preprocessing.

The programs' executing process uses the API calls. For example, in order to perform the searching of the executable files to be infected, a virus program as a rule uses the following sequence of API calls: FindFirstFileA, FindNextFileA and FindClose, which are located in the KERNEL32.DLL library. Thus, the specified sequence of API calls can be used to build the signature of the malicious program. In general, it can be noted that the usage of API calls as a signature allows isolating a constant semantic (behavioral) component, while the syntactic component will be different.

In order to trace API call, the software that monitors and displays API calls made by observed applications and services was used. A result of the data preprocessing stage is a file with a list of API calls. The next stage of the method involves the signature generation based on API call tracing.

3.2 Signature generation based on API call tracing

Signature generation of the a malware instead of using all the API calls the malware performs, involves only critical API call [9, 10]. Critical API calls contain all API calls that can lead to security infraction, changes to the operating system's behavior or API calls used for communication (modification of the system registry value, Input/Output, API functions for network resources access, etc.). It should be noted that in the process of the malware's signature generation doesn't take into account the API calls which can be added or removed from the virus program without modifying its malicious behavior (for example, MessageBox, printf, etc.).

The signature of program behavior based on API call tracing can be presented as a set of two components (fig. 1): the call frequency and the interaction of the critical API calls. An analysis of the first component allows determining the distribution of the critical API calls by groups concerning theirs malicious activity and displays the quantitative component of the signature. The second component of the signature implies the mapping the nature of the interaction of malware's critical API calls into the vector space, and have describes their interactions.

In order to describe the nature of the critical API calls interaction, let us present the malware as a directed graph:

$$G_V = \langle V, E \rangle, \tag{1}$$

where V is a set of vertexes of a graph, which presents a group of critical API functions, and E is the set of transitions between groups of the critical API functions, which describe the malwares behavior.

For a formal definition of the malware's signature, let us present it as a tuple:

$$S = \left\langle A, F, \left\langle D, d_G, n_E \right\rangle \right\rangle, \tag{2}$$

where A is the set of API calls, performed by malware of the class C_i; F is a set of frequencies of the critical API calls; D is vector of the graph's vertex degrees; d_G is the diameter of the graph; n_E – number of edges of the graph.

During signature generation of the malware's behavior based on the tracing of API calls, common to both phases is the categorization of API calls by classes. To generate the signature of malware, all set of critical API calls were divided into 26 classes [11, 12]. Table 1 shows the examples of API calls classes and their description. For example, the DeleteFiles and CreateDirectory functions are defined as Class B, i.e. functions for processing files and directories. If we have sequence API calls with CallNextHookEx, isDebuggerPresent and CreateProcess, then we will receive the following sequence "AFH" as the part of the signature. Mentioned above representation of API calls can compactly store the program's behavior presented by API calls. Additionally, combining API calls into classes of critical actions allows representing a set of functions as a group by theirs functionality (for example, CreateProcessAsUser and CreateProcess are similar by its executed functions) and can be used for different samples belonging to the same malware family. Furthermore, the process of signature generation and categorization of critical API calls doesn't take into account the function's input parameters and the result of its execution.

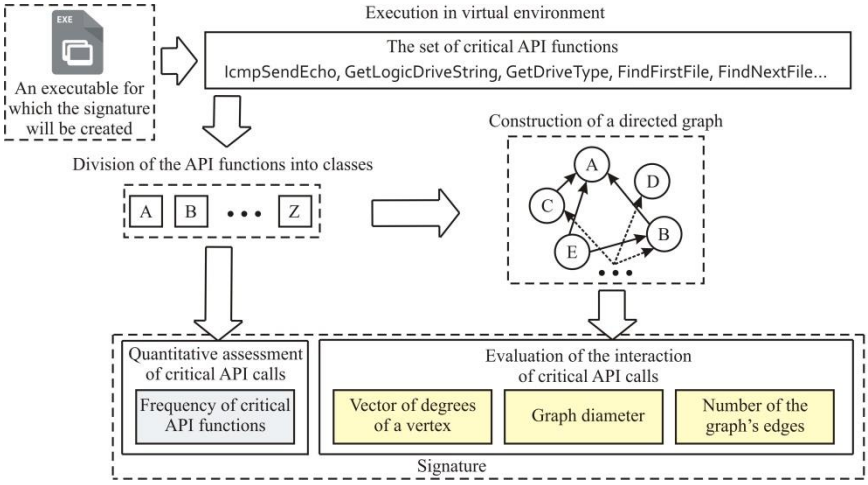

Fig.1. The signature generation of the malware's behavior based on the tracing of API calls

Table 1. API calls classes, their description and examples

API class	Description	Examples of API function	Number of API
Class A	Hooking function	CallNextHookEx, SetWindowsHookEx	12
Class B	File and directory	DeleteFiles, CreateDirectory, CopyFile	242
Class C	System registry	RegCreateKey, RegDeleteValue	48
Class D	Synchronization	CreateMutex,CreateMutexEx	213
…	…	…	…
Class Z	Device management features	DeviceControl, DvdLauncher	24

The analysis of the frequency critical API calls allows defining the membership degree - a measure of belongingness of malicious program to a malware class. It will determine the relationship between the malware sample and one of the malware classes in terms of the number of critical API calls. This is a necessity statement for assignment of the suspicious program into one of malware or benign programs classes. However, the analysis of this parameter does not provide information about the nature of the interaction between critical API calls, and, accordingly, it is not possible to refer the suspicious program to a certain modification of the malware, but only to the whole class.

Therefore, the second component of the malware's signature is intended to reflect the nature of the interaction and the relationship of the malware's critical APIs, as it allows separating the malware samples within the class.

To this end, the virus program can be represented as a directed graph D (1). For example, having the a set of degrees of vertex {3,3,2,2,1,1}, it is possible to construct the following graphs as in fig.2.

Examples of simple graphs demonstrated in fig. 2 with the same degrees of vertex are characterized by the presence of a constant component (connections in the form of a square). Similar patterns can be inherent in malware.

In addition, to distinguish malwares within the class, let us involve two features: the graph diameter and the number of edges. The graph diagram determines the maximum sequence of the critical API calls, while the number of edges determines the total number of actions performed by the malware.

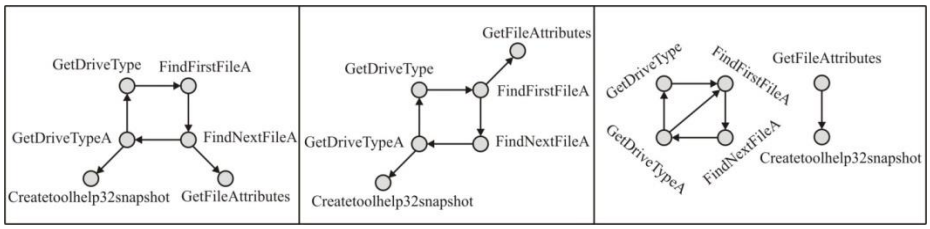

Fig. 2. Graphs with same the vertices degrees of a {3,3,2,2,1,1}
(the vertices represent the critical API calls)

3.3 Determination of the degree of membership of malicious program to a malware class.

One of the components of the proposed signature is a set of frequencies of the critical API calls. On the basis of this set the definition of the degree of membership of malicious program to a malware class is carried out. Thus, the signature base besides the frequencies of the critical API calls should contain the degree of membership of malicious program to a malware class.

The evaluation process for the degree of membership of malicious program to a malware class is based on the difference between the number of API calls performed by malicious program and benign applications in the process of their own execution. Therefore, the distinguishing between the classes of malicious programs and benign applications is possible by their behavior, that is, by the sequence of critical API calls.

In order to construct the behavior of a malware class $C_i = \{c_i^1, c_i^2, ..., c_i^x\}$ on the basis of frequencies of the critical API calls, let us represent the behavior of an malware's sample of this class as a tuple (c_i^j –malware's sample of the class C_i, where x – the number of malware's samples of the class C_i):

$$c_i^j = \langle f_1, f_2, ..., f_{26} \rangle, \tag{3}$$

where $f_1, f_2, ..., f_{26}$ – the frequencies of the critical API calls.

Let us group all the frequencies values of the critical API calls c_i^j ($\forall c_i^j \in C_i$) and represent them in the form of a matrix R_{C_i} :

$$R_{C_i} = \begin{bmatrix} c_i^1 = \langle f_1, f_2, ..., f_{26} \rangle \\ c_i^2 = \langle f_1, f_2, ..., f_{26} \rangle \\ ... \\ c_i^x = \langle f_1, f_2, ..., f_{26} \rangle \end{bmatrix}, \tag{4}$$

Based on the formed matrix R_{C_i} , let us definite the malware's behavior of the class C_i as the set of mean values of the calls for each of critical API functions class:

$$S_{C_i} = \langle F_1, F_2, ..., F_{26} \rangle, \; F_i = \frac{1}{x} \sum_{j=0}^{x} f_j, \tag{5}$$

where *j* is the class of the critical API calls.

On the basis of the received behavior of the malware class S_{C_i} , the determination of the degree of membership of malicious program to a malware class C_i is carried out using Chi-square test. The Chi-square test determines the maximum probability of a statistical significance test that measures the difference between proportions in two independent samples.

Then, to obtain the membership degree to class C_i , with the use of the Chi-square test, the determination of the difference between the proportions in the signature of the malware class S_{C_i} and each of the samples' behaviors of the c_i^j with the correction for continuity using the Yates's correction is carried out as follows:

$$\chi_j^2 = \sum_{l=1}^{26} \frac{(| c_{i,l}^j - S_{C_{i,l}} | -0.5)^2}{S_{C_{i,l}}}, \tag{6}$$

where *l* is the corresponding class of critical API calls.

As a result a set of values pairs (χ_i^2, c_i^j) is obtained.

The next stage of method involves the determining of the average value of the membership degree to the malware class C_i using formula:

$$\mu_{C_i} = \frac{1}{x}\sum_{i=1}^{x} \chi_i^2. \quad (7)$$

Thus, the parameter μ_{C_i} determines the membership degree of suspicious sample c_i^j to malware class and allows evaluating malware relationship within the class C_i.

3.4 Detection of a malicious program represented by signature of program behavior based on API call tracing.

Having the membership degrees μ_{C_i} for each of malicious and benign programs to classes C_i and the constructed signatures S_j it is possible to perform the suspicious program's detection.

The first step of the proposed method of detection involves determining the membership degree to one of the malicious or benign programs class. For this purpose, using the Chi-square test (7), the difference between the proportions of the frequencies of the critical API calls of a suspicious program (first part of S signature) and the frequency of critical API calls for each class (5) is determined.

As a result, a set of the values of membership degree of the suspicious program to each of the classes is obtained. Then the class with the best matches for the given signature is determined based on the following condition:

$$\min(|\mu_{S_j} - \mu_{C_i}|) \quad (8)$$

where μ_{S_j} is the value of the membership degree to each of the *j*-th malware classes.

As a result, a class C_i that corresponds to the suspicious program by the frequency of critical API calls is determined.

The next step of the method involves the search of the virus signature within the class C_i. Let us consider the second component of the proposed signature (2) and denote it as a features vector *V*:

$$V = \langle D, d_G, n_E \rangle = \langle v_1, v_2, ..., v_{28} \rangle \quad (9)$$

The specified vector consists of 28 numerical attributes, where 26 characters determine the degree of vertex of the graph (each vertex of the graph is determined by the class of critical calls of ARIs), and the last two are the diameter of the graph and the number of edges.

In order to distinguish the suspicious program within a malware classes, the classification of features vector *V* is carried out. It allows to assign the suspicious program to one of the virus modifications. As an algorithm of machine learning, a Naive Bayes classifier was chosen, as it is widely used in image recognition, is easy to implement and does not require a large training set [13, 14]. The idea behind a Naive Bayes algorithm is the Bayes' Theorem and the maximum posteriori hypothesis. Bayes theorem finds the probability of an event occurring given the probability of another event that has occurred already. In order to determine the belonging of the features vector *V* to

the *j*-th modification of the *i*-th malware family $C_{i,j}$ with probability $P(V \mid C_{i,j})$, let's write down Bayes' theorem in the following way.

$$P(C_{i,j} \mid V) = \frac{P(V \mid C_{i,j})P(C_{i,j})}{P(V)} \tag{10}$$

The determination of the most probable hypothesis using a posterior maximum is carried out as follows:

$$c = \arg\max_{c \in C_{i,j}} P(C_{i,j}) \prod_{i=1}^{26} P(v_k \mid C_{i,j}) \tag{11}$$

Thus, the technique of the signature formation for a malware, which is invariant to minor changes, is presented. The malware detection approach via proposed malware's signatures is proposed. It allows not only to distinguish detected malware to proper class, but also to determine its modification.

4 Experiments

In order to evaluate the effectiveness of the malware detection based on proposed method, experimental studies were conducted. For this purpose, 280 malware samples received from the VX Heavens resource [15] were used. All malware belongs to the virus families Ramnit, Gammima, Delf, Bifrose and MyDoom of various modifications (Table 2). Except for the virus programs, 74 benign applications were used, which are executable files of the operating system MS Windows© (mspaint, bfsvc, etc.). All malware samples of and utility programs were divided into: the training and testing sets. The training sample set consisted of 81 viral programs and 20 benign programs. The rest of malicious and benign samples were used to conduct the testing.

Table 2. Number of samples for each of malware families and benign applications.

		Training set		Testing set	
		Malware variant	Samples	Malware variant	Samples
Malware	Ramnit	a	12	b,c	37
	Bifrose	a	17	ae, aq, bg, bh	33
	Delf	g	13	a, d, f, h, r	41
	MyDoom	c,h	21	a, b, g, f	51
	Gammima	a	18	b, c	37
Benign	Windows app	-	20	-	54

The experiments involved the execution of all the samples and obtaining its API call sequences using API Monitor [16]. The next stage involved the behavior's base formation for all viral classes. For this purpose, for each class, membership degrees were determined using the Chi-square test (3-8). The classification results are presented in the table 3. It shows that the best detection accuracy was shown for the virus programs of the class MyDoom (96,56%) with a false positive values 3,78%. At the same time, the lowest accuracy rate of detection was seen at the level 92,74%, that defines overall accuracy of the proposed technique in the range from 92,74% to

96,56%. It should be noted that in case of wrong assignment of the malware to another modification of the same class, the result of such an experiment was considered as unsuccessful. For example, if the Delf.a virus was classified as a Delf virus class with modification b.

Table 3. Classification result

Malware	FPR	FNR	Precision	Recall	Accuracy
Ramnit	**1,25**%	8,56%	**97,25**%	**96,54**%	96,42%
Bifrose	5,12%	4,23%	92,12%	92,54%	93,80%
Delf	3,32%	4,89%	91,45%	91,23%	92,74%
MyDoom	3,78%	3,54%	94,37%	93,28%	**96,56**%
Gammima	2,74%	**3,40**%	92,76%	91,61%	93,10%

5 Conclusion

The paper presents a method for a malware's signature forming based on API call tracing. Technique allows malware detection using a proposed form of signature. The program's behavior signature based on API call tracing consists of the call frequency and the nature of the interaction of critical API calls. The detection process using the proposed signature enables to distinguish the malicious programs from benign not only by the presence of the critical API calls, but also in their interaction with each other. The experimental results showed that the effectiveness of the malware detection is up to 96.56%.

Presented technique of malware detection using a proposed form of signature has shown good detection accuracy and intended for specialists in the antivirus industry, which are engaged in the analysis of malware and support for antivirus databases. However as a majority of a dynamic approaches our method have some limitations, which are primarily related to the obfuscation and detection evasion techniques employed by the malware authors who try to develop stealth malware. In future we will concentrate to overcome this shortcoming.

References

1. McAfee Labs Threat Report. December 2018. Availabe: https://www.mcafee.com /enterprise/en-us/assets/reports/rp-quarterly-threats-dec-2018.pdf
2. Savenko, O., Lysenko, S., Nicheporuk, A., Savenko, B.: Approach for the Unknown Metamorphic Virus Detection. In: 9-th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems. Technology and Applications, Bucharest, Romania, pp. 453-458 (2017)
3. Savenko, O., Lysenko, S., Nicheporuk, A., Savenko, B.: Metamorphic Viruses' Detection Technique Based on the Equivalent Functional Block Search. CEUR Workshop, Vol. 1844, pp. 555-569 (2017)
4. Pomorova, O., Savenko, O., Lysenko, S., Nicheporuk, A.: Metamorphic viruses detection technique based on the modified emulators. CEUR Workshop, Vol. 1614, pp. 375-383 (2016)

5. Ki, Y., Kim, E., Kim, H.K.: 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, Vol. 2015 (2015)
6. Wressnegger, C., Freeman, K., Yamaguchi, F., Rieck, K.: Automatically Inferring Malware Signatures for Anti-Virus Assisted Attacks. In: Proc. of the 2017 ACM on Asia Conference on Computer and Communications Security, pp. 587-598 (2017)
7. David, O.E., Netanyahu N.S.: Deepsign: Deep learning for automatic malware signature generation and classification.: International Joint Conference on Neural Networks, pp. 1-8 (2015)
8. Ndibanje, B., Kim, K.H., Kang, Y.J., Kim, H.H., Kim, T.Y., Lee, H.J.: Cross-Method-Based Analysis and Classification of Malicious Behavior by API Calls Extraction. Applied Sciences, Vol. 9 (2), pp. 1-15 (2019)
9. Gupta, S., Sharma, H., Kaur, S.: Malware Characterization Using Windows API Call Sequences: In Proc. of the Sixth International Conference on Security, Privacy and Applied Cryptographic Engineering, (2016)
10. Lysenko, S., Savenko, O., Bobrovnikova, K., Kryshchuk, A., Savenko, B.: Information technology for botnets detection based on their behaviour in the corporate area network. Communications in Computer and Information Science, Vol.718, pp. 166-181 (2017)
11. Windows Developer Center, https://msdn.microsoft.com/en-us/windows
12. Lim, H.: Detecting Malicious Behaviors of Software through Analysis of API Sequence k-grams.: Computer Science and Information Technology, Vol. 4 (3), pp. 85-91 (2016)
13. Mansour, A.M.: Texture Classification using Naïve Bayes Classifier: International Journal of Computer Science and Network Security, Vol.18, No.1, pp. 112-120 (2018)
14. Brad, G.V.: Uses and misuses of Bayes' rule and Bayesian classifiers in cybersecurity. In Proc. of the 43-rd International Conference Applications of mathematics in engineering and economics, pp. 1-8 (2017)
15. VX Heavens Computer virus collection. Availabe: http://vx.netlux.org
16. API Monitor. Availabe: http://www.rohitab.com/apimonitor