A method for detecting false positives in procedure of malware analysis

M I Starodubov¹, A E Borshevnikov¹, S A Bystrevskii¹

¹Far Eastern Federal University, Vladivostok, Russky Island, 10 Ajax Bay, 10, 690922, Russia

E-mail maxhays@live.ru

Abstract. The paper discusses the possibility of determining false positive by the clustering method. A model of malicious software based on the "behavioral" attribute of malicious objects is described in the paper. Described is a method for studying a sample of malicious objects using a clustering algorithm based on spatial density for applications with noises - DBScan.

Keywords: Clustering; Malware; Mathematical model; Malware behavior; False positive

1. Introduction
Modern operating systems and software products are grandiose and complex products. Every day, researchers and cybercriminals discover new and new vulnerabilities that are exploited by malicious software, and many users knowingly refuse security updates, which puts their systems at risk.

Malicious software often disguise themselves as regular software. As a result, modern anti-viruses, along with malicious components, give out the fact of detection to legal software.

In connection with the above, research into new methods of detecting malware, detecting false positives of malware detection, as well as related research will never lose its relevance.

2. Clustering as a method of detecting malware
Malicious software is a broad term that refers to a part of software code, software tool, or special module that is deliberately created to damage a computer system. Often, many researchers also refer to this concept as errors of software developers, which can lead to damage. However, this statement is erroneous. By themselves, these errors cannot be called malicious, since it is impossible to prove the malicious intent of the developers who made them. However, many companies try to prevent them, as this directly affects the security of their products and opens up new opportunities for attackers.

Malware always seeks to infiltrate a computer system, inflict damage, take over control of processes, and sometimes disable the system. The purpose of the malware is quite obvious - to make a profit at the expense of the system owner. Malicious software is capable of stealing, deleting, altering data, changing the functions of a computer, and taking control of it.

There are many ways to distribute malware. Malicious software most often reaches the user's device via the Internet. It can be installed when a user browses infected websites, opens malicious email attachments, downloads hacked versions of paid programs, downloads data from unconfirmed sources, etc. Thus, system infection can occur anytime the device is connected to the Internet.
Very often, malicious software is hidden in popular applications or embedded in their updates. As a result, the likelihood of deceiving even the most attentive user increases significantly.

Clustering is one of the methods for solving the problem of detecting malware. Another concept is closely related to clustering, classification is the process of assigning an object to a certain class. Clustering differs from classification as follows:

1. The number of object groups is unknown in advance;
2. The content of the groups is unknown in advance;
3. Groups are formed based on a certain proximity of objects.

Cluster analysis can be represented as the following algorithm:

1. Formation of a set of sample objects under study;
2. Determination of the features of objects to solve the problem of drawing up vectors of characteristics;
3. Normalization of vectors of characteristics;
4. Determination of similarity for a given metric;
5. Application of the method of cluster analysis in the framework of solving the problem of partitioning into clusters;
6. Presentation of results in a form convenient for the researcher.

There are a lot of clustering algorithms. Almost all algorithms were studied in the study of the set of malicious objects. Here are the main types:

1. Hierarchical cluster analysis [1];
2. Square-error schemes [2];
3. Fuzzy algorithms [3];
4. Graph clustering algorithms [4];
5. Minimum Spanning Tree Algorithm;
6. Layer-by-layer algorithms [9].

These algorithms are currently used in a large number of problems: in detection problems, an ensemble of decision trees, locally stable convolutions are used. However, in this case, there is a possibility of a large number of false positive detection of malware.

Machine learning clustering is often more efficient than other methods [6].

### 3. Malware model

The problem of clustering malware has many solutions. Let's describe the model of malicious software.

Suppose that the source code of the program, obfuscation methods are not important to us, but only the behavior is important. In reality, the choice of a class of malware depends only on behavior, just like the choice of a particular family. This paper considers the problem of creating a method for studying anomalous points of the considered data sample.

Consider the set $U$ is the set of all possible programs:

$$U = \{PR_1, ..., PR_s, R_1, ..., R_k, T_1, ..., T_m\}, n = s + k + m,$$

where, $n$ is the number of all programs; $s$ is the number of all programs, probably not RansomWare; $k$ is the number of all RansomWare programs; $m$ is the number of programs that are not trusted by RansomWare.

Let the set $Api = \{Api_1, ..., Api_{84}\}$ is the set of Boolean functions.

Each element of the set $Api$ has the following meaning: the function $Api_i$ returns the value 1 if and only if during the execution of the program the $i$-th API-function of the system from the set $Api$ was called.

Let the set $Dll = \{Dll_1, ..., Dll_{35}\}$ is the set of Boolean functions.

Each element of the $Dll$ set has the following meaning: the $Dll_i$ function returns the value 1 if and only if during the execution of the program the $i$-th dynamic link library of the system from the set $Dll$ was called.

Let the set $File = \{File_1, ..., File_{20}\}$ is the set of Boolean functions.
Each element of the File set has the following meaning: the $File_i$ function returns the value 1 if and only if the $i$-th operation with a file from the set File took place during the program execution.

Let the set $Reg = \{Reg_1, ..., Reg_{261}\}$ be the set of Boolean functions.

Each element of the $Reg$ set has the following meaning: the $Reg_i$ function returns the value 1 if and only if during the execution of the program there has been a change in the $i$-th registry key from the set $Reg$.

Let us define any element of the set $U$ as a set of the following vectors. Each element $U_i$ from the set $U$ has the form presented in formula:

$$U_i = \left\{ \left[ \begin{array}{c} Api_{1,i} \\ ... \\ Api_{84,i} \end{array} \right], \left[ \begin{array}{c} Dll_{1,i} \\ ... \\ Dll_{35,i} \end{array} \right], \left[ \begin{array}{c} File_{1,i} \\ ... \\ File_{20,i} \end{array} \right], \left[ \begin{array}{c} Reg_{1,i} \\ ... \\ Reg_{261,i} \end{array} \right] \right\}, i = 1, n \quad (2)$$

Let $\varphi(U_i): U_i \rightarrow \{0, 1\}$ be a functional denoting the execution of the program $U_i$ and leading either to a safe state of systems (0) or an unsafe state (1). Let us define the area of probabilities of malicious software behavior as $V = \{U_i: U_i \in U, P(\varphi(U_i) = 1)\}$.

Let us define the sets $G = \{PR_1, ..., PR_s, T_1, ..., T_m\}$ and $R = \{R_1, ..., R_k\}$.

The selection of the sets $G$ (a set of programs not verified by analysts) and $T$ (a set of programs tested by analysts) from the set $G$ is conditional and is set only for the convenience of virus analysts.

4. Experiment

As the set under study, a sample of 994 files with the following properties was formed:

1. All files are executable objects in the operating system Windows, i.e. have a PE format;
2. The files must have a detection verdict of the type Trojan-Ransom.Win32.Agent;
3. Files must be emulated by means of Cuckoo Sandbox.

Popular resources for storing and analyzing malicious objects were selected as a source of objects: VirusTotal, VirusShare.com, Malware.lu, etc.

The study of the executable code of each object in modern realities often takes a lot of analyst’s time and effort. In this work, a different approach was used, a behavioral model of malicious objects was investigated and developed.

The following configuration was chosen as an environment for investigating the behavior of objects: Cuckoo Sandbox, a virtualization product of Oracle VM VirtualBox with a virtual machine on which operating system Windows 7 is installed and popular libraries (vc++, .net and others) are preinstalled.

This configuration was chosen for the following reasons:

1. Ability to connect additional analysis systems to Cuckoo Sandbox, including traffic analysis systems;
2. Cuckoo Sandbox is an open source project, which is why the source code is available for study and it is possible to change it for a specific one;
3. Operating system Windows 7 was chosen as a guest operating system, as it is the second most Internet-users system of the Windows family, but at the same time it is much less secure than operating system Windows 10.

During the analysis, the following actions take place:

1. The executable object is sent for analysis;
2. A virtual machine with known parameters is launched on the physical computer and the object being analyzed is passed to it for execution;
3. The parameters changed after the execution of the object are transferred back to the physical computer;
4. A behavioral report of the object is generated.

A behavioral report is one of the most important parts of an analyst’s analysis of malicious objects, along with the source code, and can give a complete picture of the hidden capabilities of the object under investigation.
The formation of a vector of characteristics is the starting point for solving the clustering problem. As a result, from the available data of the Cuckoo Sandbox report and the typical behavior of malicious objects, the most important characteristics for analysis were identified, namely, important API functions [7], important registry keys, calls to important dynamic link libraries, changes to important system files and directories [8].

Each characteristic of a vector is a boolean value:
- 0 – the object did not call the API function, did not change the registry key, etc.
- 1 – performed one of the above actions.

Each vector of characteristics is saved to the database of all analyzed objects. As a result, a base of vectors of characteristics is formed. After adding another vector, a new object is sent for analysis. The local csv table is temporarily used as the base used.

Cluster analysis works best on a set of normalized vectors. This problem was solved at the stage of forming the vector of characteristics. Since the set of values for each of the characteristics is clearly defined and discrete, it is easy to check that each value falls into the segment [0;1].

As can be seen in figures (figure 1, figure 2), the reduction of vectors to the same length is not only meaningless, but also harmful [9] within the framework of solving our problem. After bringing the vectors to the same length, the possibility of finding anomalous points is lost.

The DBSCAN method was used to perform the clustering task. The implementation of this method within the sklearn library takes the following parameters as input:

1. eps - maximum distance to the nearest object of one cluster;
2. min_samples - the minimum number of objects in the cluster. Experimentally, it was found that the optimal maximum distance to the nearest object of one cluster in the frame of work with the current sample is 25.4.

From the analysis of anomalous points, it became clear that the minimum number of objects in the cluster should be set to 3.

After execution, the DBSCAN function returns the original set of data under investigation with the added characteristic labels, each $i$-th element of which indicates the belonging of the $i$-th object to a specific cluster. If labels has a value of -1, then such a point is considered noise, and the object will be called anomalous.

One of the simplest and most convenient methods for visualizing clustering results is a projection onto a two-dimensional plane. As part of the study, it was decided to use the principal component analysis to reduce the dimension of the parameters under study.

As can be seen in figures (figure 3, figure 4), the shapes of the clusters are quite clearly visible, and also, what is more important in the framework of our work, anomalous points are visible (marked in black).

Let a point that does not belong to any cluster be called anomalous.

Analysis of anomalous points is the key point of this work. The points that get out of the general picture of the family's behavior are obviously not in their own class, family, or are very specific
representatives of this family. In our sample of data for anomalous points, the following options are possible:

1. A specific representative of his family;
2. False positive detection of malware;
3. Incorrect class, family of the fact that malware was detected.

Analyzing the data sample, 14 anomalous points were identified.

5. Conclusion
Here are the main results of the study:
1. The proposed method shows its productivity;
2. All anomalous points found are objects either mistakenly assigned to the Trojan-Ransom.Win32.Agent class, or specific representatives of this class;
3. This method depends on the professionalism of the analyst who performs the manual analysis.

The results obtained within the framework of the study can be applied to modify algorithms for the behavioral analysis of malware.

References
[1] Sisaat K, Kittitornkun S, Kikuchi H, Yukonhiatou C, Terada M and Ishii H 2016 A Spatio-Temporal malware and country clustering algorithm: 2012 IJ MITF case study Int. J. Inf. Secur. 16 pp 459-473
[2] Cordeiro de Amorim R and Komisarczuk P 2014 Partitional Clustering of Malware Using K-Means Cyberpatterns Unifying Design Patterns with Security and Attack Patterns pp 223-233
[3] Pai S, Di Troia F, Corrado V A, Austin T H, Stamp M 2016 Clustering for malware classification J. Comput. Virol. Hack. Tech. 13 pp 95-107
[4] Chen Y, Liu F, Shan Z and Liang G. 2018 MalCommunity: A Graph-Based Evaluation Model for Malware Family Clustering 4th Int. Conf. of Pioneering Computer Scientists, Engineers and Educators (Zhengzhou) part 1 pp 279-297
[5] Kinable J and Kostakis O 2011 Malware classification based on call graph clusterin J. in Computer Virology 7 pp 233–245
[6] Narra U, Di Troia F, Corrado V A, Austin T H and Stamp M 2015 Clustering versus SVM for malware detection. J. Comput. Virol. Hack. Tech. 12 pp 213–224
[7] Szeles G, Cole A 2019 Malware Clustering Based on Called API During Runtime Information and Operational Technology Security Systems pp 110–121
[8] Ashraf A, Aziz A, Zahoora U, Khan A 2019 Ransomware Analysis using Feature Engineering and Deep Neural Networks arXiv: Cryptography and Security
[9] Milosevic N and Huang J 2019 Deep learning guided Android malware and anomaly detection arXiv: Cryptography and Security