Constructing Heuristic Malware Detection Mechanism Based on Static Analysis

Alexander Kozachok

***Abstract*— To ensure the protection of information processed by computer systems is currently the most important task in the construction and operation of the automated systems. The paper presents the application justification of a new set of features distinguished at the stage of the static analysis of the executable files to address the problem of malicious code detection. In the course of study, following problems were solved: development of the executable files classifier in the absence of a priori data concerning their functionality; designing class models of uninfected files and malware during the machine learning process; development of malicious code detection procedure using the neural networks mathematical apparatus and decision tree composition relating to the set of features specified on the basis of the executable files static analysis. The paper also describes the functional model of malware detection system using the executable files static analysis. The conclusion contains the results of experimental evaluation of the developed detection mechanism efficiency on the basis of neural networks and decision tree composition. The obtained data confirmed the hypothesis about the possibility of constructing the heuristic malware analyzer on the basis of features distinguished during the static analysis of the executable files. However, the approach based on the decision tree composition enables to obtain a significantly lower false negative rate probability with the specified initial data and classifier parameter values relating to neural networks.**

***Tóm tắt*— Trong hệ thống mạng, vấn đề an toàn và bảo mật thông tin đóng một vai trò hết sức quan trọng. Bài báo này trình bày cơ sở lý thuyết về khả năng sử dụng không gian đặc trưng mới, được trích chọn trong quá trình phân tích tĩnh các tập tin thực thi để giải quyết bài toán nhận diện mã độc. Đóng góp khoa học trong bài báo bao gồm: Xây dựng bộ phân lớp tập tin thực thi trong trường hợp thiếu các thông tin tiên nghiệm, mô hình hóa lớp các tập tin bị lây nhiễm và phần mềm độc hại trong quá trình học máy; Xây dựng phương thức phát hiện phần mềm độc hại sử dụng mạng nơron và cây quyết định. Bài báo cũng mô tả mô hình hệ thống phát hiện phần mềm độc hại bằng cách sử dụng phương pháp phân tích tĩnh các tập tin thực thi. Phần kết luận trình bày các kết quả thực nghiệm để đánh giá hiệu suất mô hình phát hiện đã được đề xuất. Các kết quả thu được đã khẳng định giả thuyết về khả năng xây dựng công cụ phân tích phần mềm độc hại sử dụng phương pháp heuristic dựa trên các đặc trưng được trích chọn trong quá trình phân tích tĩnh các tập tin thực thi.**

***Keywords*: anti-virus protection; malware; neural networks; decision trees; heuristic analysis; machine learning; detection.**

***Từ khóa:* ngăn ngừa virus; phần mềm độc hại; mạng nơron; cây quyết định; phân tích heuristic; học máy; phát hiện.**

I. INTRODUCTION

Security of information processed by computer systems poses the most important task for building and operating of automated systems today. Along with that, one of the most dangerous threat is computer malware that can modify (delete) user data, steal confidential information, slowdown or disable operating system. This research substantiates the possibility of using feature space based on the static analysis of executable files for solving the task of heuristic malware detection using the neural networks and decision trees composition mathematical apparatus.

Today there exist different malware detection techniques. The most widely known techniques are the following ones [1]:

- signature-based search (malicious code detection based on byte sequence which definitely characterizes it);
- heuristic search (code detection based on indirect attributes which characterize it as being malicious);
- behavioral mechanisms that affect executing forbidden operations by different processes (e.g., access to critical memory areas or executable code injection into other processes).

All above-listed techniques have essential weak points, i.e. they possess limited capabilities

This manuscript is received on 15/02/2017. It is commented on 20/2/2017 and is accepted on 20/3/2017 by the first reviewer. It is commented on 25/2/2017 and is accepted on 30/3/2017 by the second reviewer.

for detection of modified and new viruses, or require the user to be involved in the decision-making process with respect to file belonging to a certain class.

Today the antivirus software cannot guarantee 100% malware protection. The results of tests performed by AV-Comparatives in March 2017 show that heuristic detection rate of new malware strains amounts to approximately 90-95%, whereas false positive rate is about 1-3% for most of the modern antivirus software [2].

To detect new malware, heuristic methods or more generally statistical approaches are the most promising research trends nowadays. Some of them based on structural analysis and executable file features [3-5]. One of the solutions for increasing the effectiveness of heuristic malware detection process is the development of new tools and techniques for malware detection. The purpose of this study is to prove the possibility to build a heuristic technique for malware detection based on static analysis of executable files. The distinctive feature of the suggested approach consists in the use of new feature space for building a heuristic detector based on the well-known machine learning techniques, i.e. neural networks and decision trees composition. In this context, a decision on the malware presence will be taken according to a certain law based on availability or absence of totality of features from criteria array defined at the stage of executable file static analysis.

II. FORMING FEATURE SPACE BASED ON STATIC EXECUTABLE FILES ANALYSIS

In order to substantiate the possibility of using suggested feature space for solving the task of heuristic malware detection, the neural networks and decision trees composition technique has been applied in this study. Let us consider the totality of the features being studied.

The whole feature space may be divided into eight conditional feature groups. Group 1 comprises the features based on the results of characteristics evaluation for the following executable files parts: file header size, optional header, MS-DOS header, digital certificate. Since the structure of the headers has been defined, in case of their size change relevant attribute will be detected. Group 2 comprises the features associated with the use of packing, archiving and encrypting utilities for executable files such as UPX, MPRESS, PeCompact etc. Group 3 comprises information about dynamic libraries, as well as functions exported and imported by the executable file. The rate of certain API-functions and dynamic libraries usage by malware has been precomputed. As a result, two classes have been identified. The first class comprises API-functions by means of which malicious actions can be performed. The second class comprises the rest of the functions. In this context, belonging to a certain class of API-functions is to be regarded as a feature. Group 4 comprises data on digital certificates, namely, whether they are available in the file, whether data are out-of-date or have been recalled. Group 5 comprises the features based on the information about PE-file structure, namely, availability of anonymous section, whether “overlay” technique is applied, whether the first section is available for writing, whether the control function is transferred by the file to other files, entry point address is out of the file section boundaries, in the first or other sections, whether the last and other sections are of executable type. Feature group 6 is formed based on the manifest, its availability, correspondence of the manifest structure to standard format, whether the administration privilege is requested by the manifest etc. Group 7 comprises information about executable file interface with the operating system, namely, whether the use of Structured Exception Handling (SEH), Data Execution Prevention (DEP) and Address Space Layout Randomization (ASLR) is ignored, whether the application is executed in Visual Basic virtual machine, whether Thread Local Storage (TLS) is used. Group 8 comprises information not included in previous seven groups; for example, whether the file contains rigidly fixed IP-addresses, whether direct cookie links are present, whether databases are in use etc.

Each analyzed executable file is described in the form of a Boolean vector, where one means the feature is available, zero means no feature is available.

III. STATIC HEURISTIC MALWARE DETECTION MECHANISMS

A. Neural malware detection mechanism based on static executable files analysis

To increase the effectiveness of heuristic analysis of executable files we suggest using the malware detection technique based on neural networks [6-9]. Utilization of the neural network mathematical apparatus together with the created feature space will enable us to solve the following tasks:

1) generating class models in the course of learning (uninfected files and malware);

2) developing malware detection procedure through the use of feature vector based on static analysis of executable files;

3) classifying executable files without priori data on their infection with malicious code.

Solution of the task for developing the neural network malware detection technique based on static analysis of executable files comprises two main stages as follows:

1) learning the neural network which defines malware and uninfected file classes (learning subsystem);

2) calculating output values of neural network based on the sequence of features singled out of the files analyzed, and decision-making on files belonging to a certain class (classification subsystem).

Figure 1 shows the functional model of malware detection system based on neural networks.

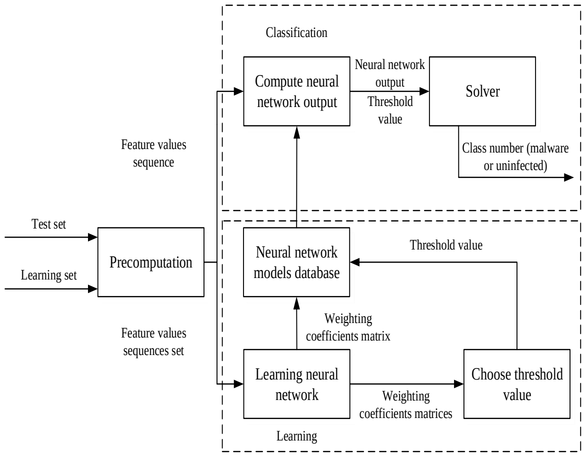

Classification
Compute neural network output
Neural network output
Threshold value
Solver
Class number (malware or uninfected)
Feature values sequence
Test set
Learning set
Precomputation
Neural network models database
Threshold value
Weighting coefficients matrix
Feature values sequences set
Learning neural network
Weighting coefficients matrices
Choose threshold value
Learning

Figure 1. Functional model of malware detection system based on neural networks.

Let us consider the learning process in detail. It consists of two main stages:

1) supervised learning the single-layer neural networks (single-layer neural networks were chosen just for test purposes to decrease computational complexity);

2) choosing the threshold value for decision-making on the file infection status.

At the first stage, the neural network is learnt in supervised mode. The network is provided with values of both input and priori known output signals, whereas weight coefficients are subject to corrective adjustment in order to increase the accuracy of the neural network learning.

The result of the first learning stage is a weighting coefficients matrix of the learnt neural network for each file class.

At the second learning stage for the given learning set it is necessary to evaluate mistake probability values of the first and second grade depending on the threshold value, and choose threshold value in accordance with the requirements to the first and second grade mistake criticality for further effective functioning of the neural network as a malware detection tool.

As a result of the second learning stage a threshold value to be used for executable file classification needs to be chosen. The learning procedure result is a weighting coefficients matrix and threshold value.

The detection procedure consists of two main stages [10, 11]:

1) calculation of initial values for neural networks;

2) decision-making on the file belonging to a certain class.

During the first stage the detection system input receives the feature sequence singled out of the file analyzed. Then the neural network output values are calculated using weighting coefficients matrices entered into the database.

During the second stage the obtained value is compared with the threshold one and decision on the analyzed file belonging to a certain software class is taken.

B. Heuristic malware detection mechanism based on decision trees composition

As an alternative approach, in order to substantiate the possibility to build heuristic malware detection tool based on static analysis of executable files this study provides the results of classifier effectiveness evaluation based on the decision trees composition.

As a rule, composition of algorithms is regarded as a combination of N algorithms $d_1(x),...,d_N(x)$ in a single one. The idea consists in learning the algorithms and averaging of the obtained responses:

$$a(x) = \frac{1}{N}\sum_{n=1}^{N} d_n(x)\,. \qquad (1)$$

This formula directly answers the regression problem. For the case of binary classification $d_1(x),\ldots,d_N(x)$ it is necessary to take a sign from the resulting formula:

$$a(x) = \operatorname{sign}\frac{1}{N}\sum_{n=1}^{N} d_n(x)\,. \qquad (2)$$

To build a decision trees composition [12] first it is necessary to learn the basic *N* algorithms on different subsets singled out from the learning set. To single out random sets, an approach based on the random sets generation from the learning set through removal followed by return procedure (bootstrap) has been applied in this study. At the same time, the size of each subset amounts to $0{,}632L$, where L is the learning set size.

Additionally, random subspace technique [13] has been applied. The technique consists in choosing the random subset of features for learning each basic algorithm. The number of the features chosen is a hyperparameter of the given technique.

At the learning stage for both approaches described above it is necessary to create two representative learning sets: uninfected files and malware. Test set is to be created using the files that are not included in the learning file set. During the pre-processing stage, a totality of features is singled out in the form of a Boolean vector from the executable files sent to the system input.

For learning block it is required to single out a totality of feature sequences from the whole totality of files of the representative learning set. For detection block it is required to single out a feature sequence from one file which was received at the classification system input.

Initial data used in the study:

- 1862 uninfected files (system and program files collected from different Windows operating systems);
- 1910 malware files (authors private collection);
- 353 features singled out based on static analysis of executable files.

As a result of performed static analysis of provided file sets, a feature vector has been generated for each executable file of both classes.

The rated evaluations of features distribution for uninfected file class and malware class are shown in Figures 2 and 3 respectively.

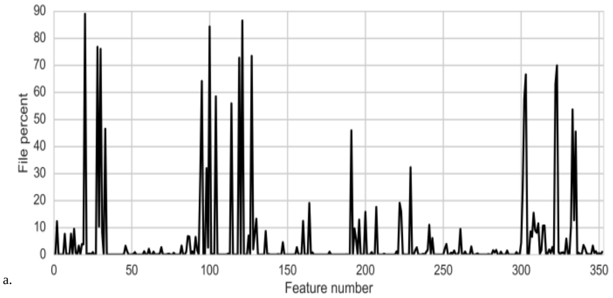


Figure 2. Feature distribution estimation for uninfected files.

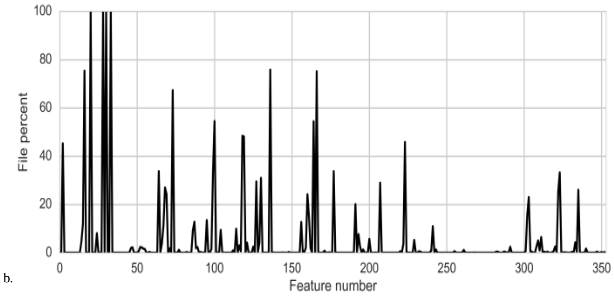


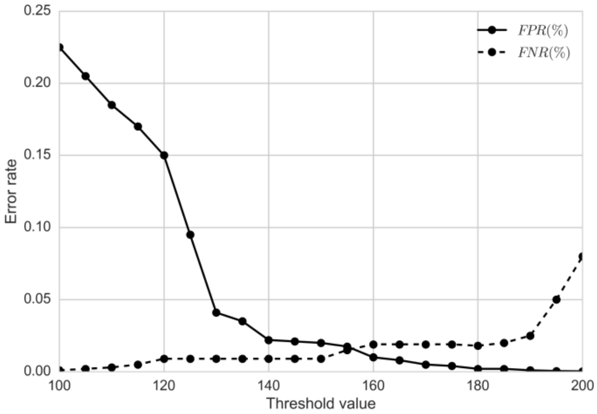
Figure 3. Feature distribution estimation for malware.

The provided diagrams cannot give the basis for making firm conclusion on statistic difference of both classes within the framework of created feature space. However, the groups of performed experiments have confirmed the hypothesis that building a malicious code heuristic analyzer based on the static analysis of executable files is possible.

To evaluate the effectiveness of the neural network technique the following initial data have been used:

- learning rate factor 0.1;
- accuracy 0.001;
- number of iterations for reaching the required learning accuracy has been limited by 1000.

Figure 4 shows the results of experimental evaluation of the malware detection technique software implementation based on neural networks: mistake probability of the first (*FPR*) and second (*FNR*) grade depending on the threshold value parameter.



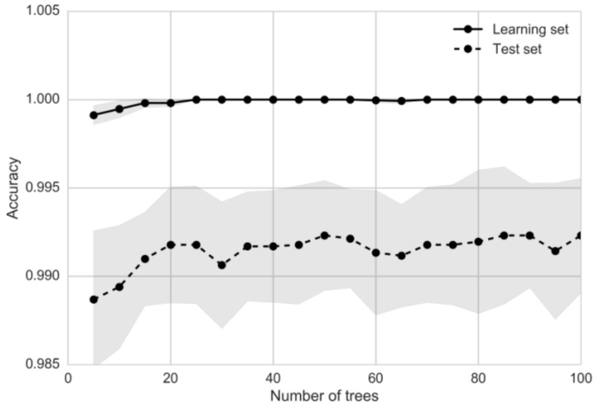
c.

Figure 4. Error rate distribution depending on threshold value.

The performed groups of experiments have enabled to substantiate the threshold value of the hyper parameter of the malware detection neural network technique equal to 180. As a result of the developed malware detection technique application the following mistake probability values have been obtained:

- first grade mistake probability (false positive rate) is 0.002;
- second grade mistake probability (false negative rate) is 0.018.

Then the classifier has been learnt based on the decision trees composition, and the obtained result has been cross validated. Figure 5 shows the relation between detection accuracy and number of decision trees. Along with that, the sets of both classes have been divided with the proportion of 0.7 (learning), 0.3 (test).

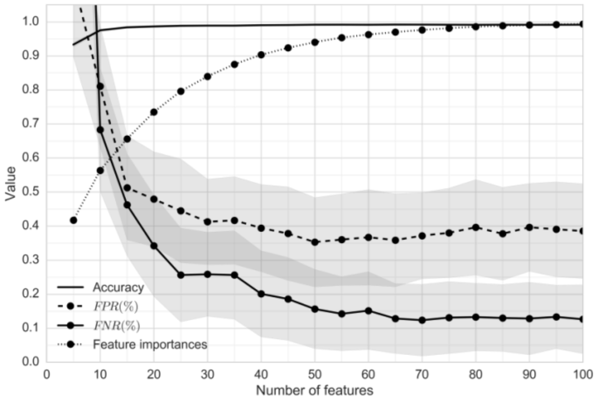


d.

Figure 5. Accuracy value depending on number of trees.

The performed experimental evaluation of the developed solution allows to substantiate the following hyperparameters of the classifier:

- number of decision trees is 40;
- number of valuable features is 50 (see Figure 6).



e.

Figure 6. Accurarcy value depending on number of features.

The following mistake probability evaluation values have been obtained when choosing the given parameters of the classifier:

- first grade mistake probability (false positive rate) is 0.0035;
- second grade mistake probability (false negative rate) is 0.0015.

IV. CONCLUSION

It should be noted that both approaches have confirmed the hypothesis that building a malicious code heuristic analyzer based on features singled out during static analysis of executable files is possible. However, the approach based on the decision trees composition allows obtaining much less probability value of false negative rate relative to neural network tool with the above-mentioned initial data and classifier parameter values.

The obtained values of mistake probabilities for developed prototypes comply with the requirements of the Federal Service for Technical and Export Control [14] imposed to antivirus software.

In conclusion it should be noted that the suggested approach consisting in the application of the space of features singled out from the executable files at the stage of static analysis and well known machine learning techniques enables us to implement a new mechanism for heuristic detection of malicious code which provides the possibility to reveal new and modified malware samples.

REFERENCES

[1] Kozachok, A. V. "Mathematical model of recognition destructive software tools based on hidden Markov models", A.V. Kozachok, "Vestnik SibGUTI", Vol. 3, pp. 29-39. – (in Russian), 2012.

[2] AV-Comparatives (2017) Malware protection test. URL https://www.av-comparatives.org/wpcontent/uploads/2017/04/avc mpt 201703_en.pdf.

[3] Shabtai A, Moskovitch R, Elovici Y, Glezer C "Detection of malicious code by applying machine learning classifiers on static features: A state-of-the-art survey". Information Security Technical Report 14(1) pp.16–29, 2009.

https://doi.org/10.1016/j.istr.2009.03.003

http://www.sciencedirect.com/science/article/pii/S136341270900004.

[4] Santos I, Devesa J, Brezo F, Nieves J, Bringas PG "Opem: A static-dynamic approach for machine-learning based malware detection". In: International Joint Conference CISIS12-ICEUTE 12-SOCO 12 Special Sessions, Springer, Springer-Verlag Berlin Heidelberg, Ostrava, Czech Republic, pp 271–280, 2013.

[5] David B, Filiol E, Gallienne K , "Structural analysisof binary executable headers for malware detection optimization". Journal of Computer Virology and Hacking Techniques Vol. 13(2),pp. 87–93, 2017.http://dx.doi.org/10.1007/s11416-016-0274-2.

[6] Uossermen, F. "Neurocomputing machinery", F. Uosserman – Moscow: "Mir" Publisher (in Russian), p.184, 1992.

[7] Smagin, A. A. "Intelligent information systems / A.A. Smagin, S.V. Lipatova, A.S. Mel'nichenko", Ul'janovsk: "UlGU" Publisher, (in Russian), p.136, 2010.

[8] Bayer, U. Scalable, "Behavior-Based Malware Clustering / U. Bayer, P. M. Comparetti, C. Hlauschek, C. Kruegel, E. Kirda", NDSS, Vol. 9, pp. 8–11, 2009.

[9] Hinton, G. E. "A fast learning algorithm for deep belief nets", G. E. Hinton, S. Osindero, Y. W. Teh Neural computation. Vol. 18(7), pp. 1527–1554, 2006.

[10] Moser, A. "Limits of static analysis for malware detection", A. Moser, C. Kruegel, E. Kirda Twenty-Third Annual Computer Security Applications Conference. pp. 421-430, 2007.

[11] Srivastava, N. "Dropout: a simple way to prevent neural networks from overfitting", N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov. – Journal of Machine Learning Research, Vol. 15(1), pp. 1929-1958, 2014.

[12] Schmind, H. "Probabilistic part-ofispeech tagging using decision trees", H. Schmind, In New methods in language processing. Routledge Publisher, p.154, 2013.

[13] Shi, T. "Unsupervised learning with random forest predictors", T. Shi, S. Horvath, Journal of Computational and Graphical Statistics Vol. 15(1), pp. 118–138, 2006.

[14] Federal Service for Technology and Export Control, "Informacionnoe soobshhenie ob utverzhdenii trebovanij k sredstvam antivirusnoj zashhity" (in Russian) 240/24/3095, 2012.

AUTHORS PROFILE

PhD. Alexander Kozachok

Workplace: The Academy of Federal Guard Service of the Russian Federation.

Email: alex.totrin@gmail.com

The education process: has received Ph.D. degree in Engineering Sciences in Academy of Federal Guard Service of the Russian Federation in Dec. 2012.

Research today: Information security; Unauthorized access protection; Mathematical cryptography; theoretical problems of computer science.