UDC – 004.725:004.852

# RESEARCH THE MODEL OF DETECTION POLYMORPHIC MALWARE BY THE CONVOLUTIONAL NEURAL NETWORK

**Timur V. Jamgharyan**
National Polytechnic University of Armenia
105, Teryan St., 0009, Yerevan
t.jamgharyan@yandex.ru
ORCID iD: 0000-0002-9661-1468
Republic of Armenia

https://doi.org/10.56243/18294898-2023.3-10

**Abstract**

The paper presents the results of research on the use of a convolutional neural network to detect polymorphic malware. The research was conducted on on basis of polymorphic software abc, cheeba, december_3, stasi, otario, dm, v-sign, tequila, flip.

The generated of datasets for training a convolutional neural network was carried out using «state matrices» of various dimensions. The Fadeev-Leverrier method was used as a mathematical apparatus. The simulation of the developed software at different iterations and visualization of the results was carried out.

***Keywords:*** machine learning, polymorphic malware, intrusion detection system, «state matrices», Fadeev-Leverrier method.

## Introduction

The heterogeneity of networks, equipment, software and other components increase the «attack surface» [1] on the network infrastructure (NI). The research [2] formulated the main conditions for creating a secure NI. In particular, it was noted that one of the most important components of the NI is an intrusion detection system (IDS) capable of adapting to possible and probable threats.

This requirement appeared after IDS «skipped» a lot of attacks that were carried out by attackers using *0-day vulnerabilities* and/or various exploits [3]. Machine learning has made it possible to transfer the means of both defense and attack to a qualitatively new level [4,5]. The increase in machine-to-machine interaction (Machine-to-Machine, M2M) has led to the fact that the very paradigm of the concept of a *secure automated system* has changed [6]. Among the many types of malware, polymorphic[1] malware is one of the most threatening. A distinctive feature of polymorphic malware from obfuscated malware is that polymorphic software changes its source code on its own, without human intervention [8]. The relevance of the work is determined by the ever-increasing role of using machine learning to attack critical Infrastructure objects using various types of malware.

[1] *Polymorphic malware - is malicious software that is characterized by the following behavior: encryption, self-propagation, and modification of one or more components of the source code* [8].

.

An important task in the development of IDS with machine learning is the generation of data sets for training neural networks that are part of the IDS. Various researchers use various methods for the analysis of «state matrices» and the subsequent generation of data sets based on them, in particular: the method of Krylov-Danilevsky, Samuelson, Hessenberg, Lanczos. These methods allow solving particular problems within the given constraints. But these methods are not suitable for constructing «state matrices» of polymorphic malware, due to the small dimension of both the matrix itself and its single element. As indicated in [9], these methods do not provide reliable deployment of determinants in systems with a high-order state vector, which is critical when building «state matrices» based on the source code of polymorphic software. Scientific novelty lies in the application of the Fadeev-Leverrier[2] method for generating «weight» coefficients and forming data sets when initializing a neural network based on a «state matrix» obtained from polymorphic malware. The application of the method allows, with an increase in the bit depth of the matrix, to reduce the amount of necessary polymorphic malware to activate the layers of the neural network. A convolutional neural network (CNN) with a different number of layers and elements was used as a neural network.

**Discussion**

1. Let there be data sets of network traffic $\{g_1, g_2, g_3 ... g_n\} \in G$ with embedded malicious polymorphic software $\{a_1, a_2, a_3 ... a_n\} \in A$.
   It is necessary to determine the ratio *A(G)* under which it is possible to unambiguously detect and identify *a.*

2. There is a state matrix $M = \begin{pmatrix} a_{11} & a_{12} & g & a_{1n} \\ a_{21} & a_{22} & g & a_{2n} \\ g & g & g & g \\ a_{n1} & a_{n2} & a_{n3} & a_{nn} \end{pmatrix}$, where - *a* is a set of malware bytes that form a data set for training, *g* - datasets of network traffic fragmented according to the rules set on the network interface. $k` = \{k_1, k_2, g, k_n\}$ matrix eigenvector *M,* eigenvalue $Mk` = \lambda k`$, where $\lambda$ eigenvalue $k`$.

3. Characteristic matrix $C = M - \lambda E \begin{pmatrix} a_{11} - \lambda & a_{12} & g & a_{1n} \\ a_{21} & a_{21} - \lambda & g & a_{2n} \\ g & g & g & g \\ a_{n1} & a_{n2} & a_{n2} & a_{nn} - \lambda \end{pmatrix}$,

where E- identity matrix (In this case, a dataset consisting entirely of malware).

With the coordinate notation of the vector $k$

[2] *The Faddeev-Leverrier method is a modified Leverrier method that allows, when calculating the coefficients of the characteristic polynomial, to calculate the eigenvectors of matrices, the inverse matrix and the matrix trace. The method requires a larger number of operations, but unlike other methods, it allows you to work with matrices of large dimensions (including elemental ones).*

$$(a_{11}-\lambda)k_1 + a_{12}k_2 + g + a_{1n}k_n = 0,$$
$$a_{21}k_1 + (a_{22}-\lambda)k_2 + g + a_{2n}k_n = 0,$$
$$g$$
$$a_{n1}k_1 + a_{n2}k_2 + g + (a_{nn}-\lambda)k_n = 0$$

The determinant C is an n[th] degree polynomial with respect to $\lambda$ $det\,\mathrm{C} = (-1)^n\left(\lambda^n - c_1\lambda^{n-1} - g - c_{n-1}\lambda - c_n\right)$ is called the characteristic polynomial.

The roots of this polynomial are the eigenvalues of the matrix *M.* To solve this problem and determine the activation function of a convolutional neural network in this study, the Fadeev-Leverrier method is used.

**Conflict Setting**

It is necessary to develop an algorithm that configures a convolutional neural network to perform a given function.

**Boundary conditions**

- the dimension of the «state matrix» is finite (≤4096 bit),
- the source code of polymorphic software (or at least 10-12% of its fragments) is known,
- at a given point in time, it is possible to search for only one type of polymorphic software.

The developed algorithm is shown Fig. 1.

**Algorithm operation**

The input of the software that searches for polymorphic software receives datasets generated both on the basis of the known code of polymorphic software and obtained by reverse engineering methods using *IDA Pro, Ghidra* tools [10].

step 1 preparation of datasets for the formation of the state matrix,

step 2 formation of the parameters of the «matrix of states» - dimension, numerical values of the elements of the matrix. The developed software is a dynamic array, which, in the presence of an appropriate computing resource, makes it possible to operate with matrices of dimension 16x16, 32x32, 64x64, 128x128, 256x256, 512x512, 1024x1024, 2048x2048, 4096x4096 elements and the size of each element $\leq 2^{12}$ bit,

step 3, 4 segmentation of the input data sets and activation of the «state matrix» of the required dimension,

step 5, 6 calculation of the value of the hash function from the data sets entered into the «matrix of states». The hash function is calculated by calling the hashlib library (step 6),

step 7 calculation of the coefficients of the «state matrix» by the Fadeev-Leverrier method,

step 8, 9 calculation of the hash value of the *sha-1* function from the datasets input to the CNN. The hash function is calculated by calling the hashlib library (step 9),

step 10 comparison of hash function values from datasets entered into the «state matrix» and hash functions from datasets entered into a CNN,

step 11 a decision-making cycle on the input of polymorphic software data into a CNN.

The main task of a CNN is: based on fragments of the source code of polymorphic software, to detect possible and probable subsequent modifications of this software. The choice of using a convolutional neural network is due to the fact that in a convolutional neural network the transfer

function (activation function of neurons) can take on a wide range of values, which is fundamentally important in the study of polymorphic software, since the range of changes in polymorphic software is a priori unknown.

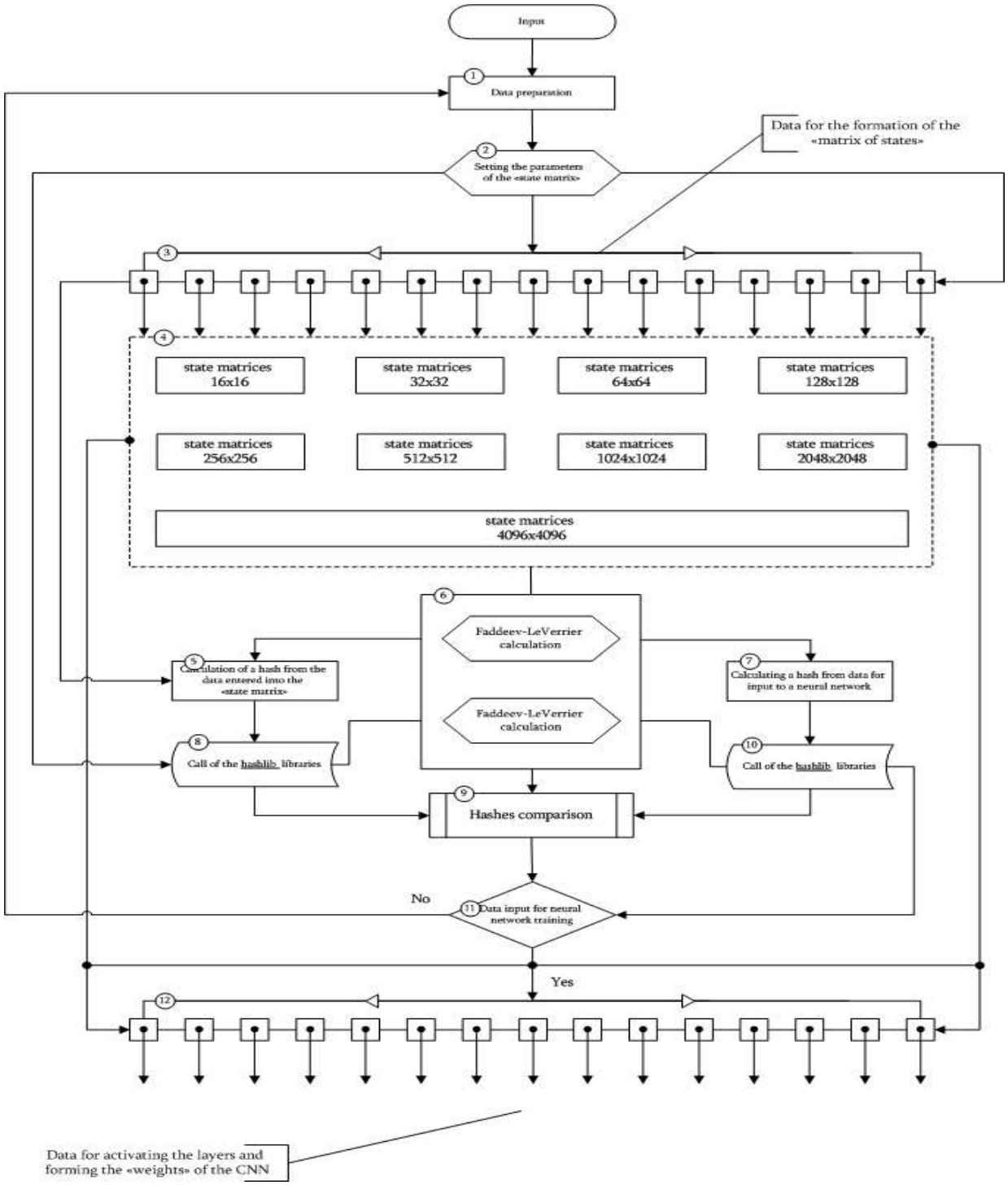


**Fig. 1 The developed software algorithm**

Visualization was carried out using the *TensorBoard* plugin [11].

**Research Results**

With an increase in the dimension of the «state matrix», the accuracy of detecting polymorphic software increases, but it also requires a large computing resource (Fig. 2,3). With an increase in the

dimension of the «state matrix», the number of program execution errors increases, which is detected and blocked, but forces a repeated search for malware, which, with large volumes of processed traffic, will require significant computing resources. Program execution errors are detected and blocked.

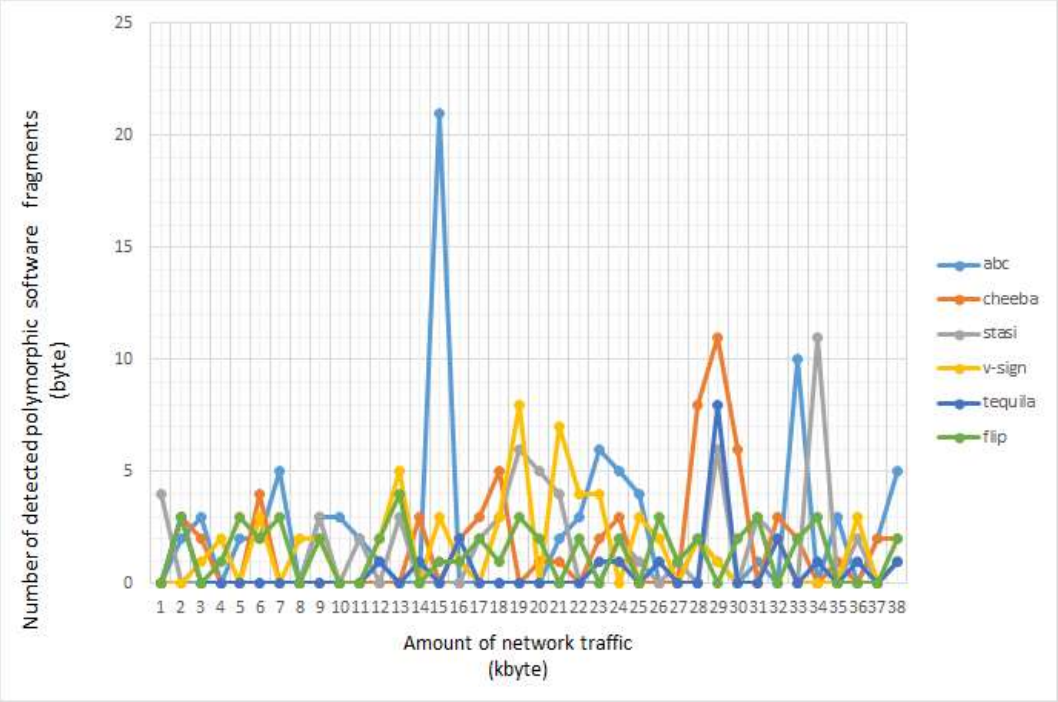


**Fig. 2 Visualization of malware detection at size «state matrices» 32x32**

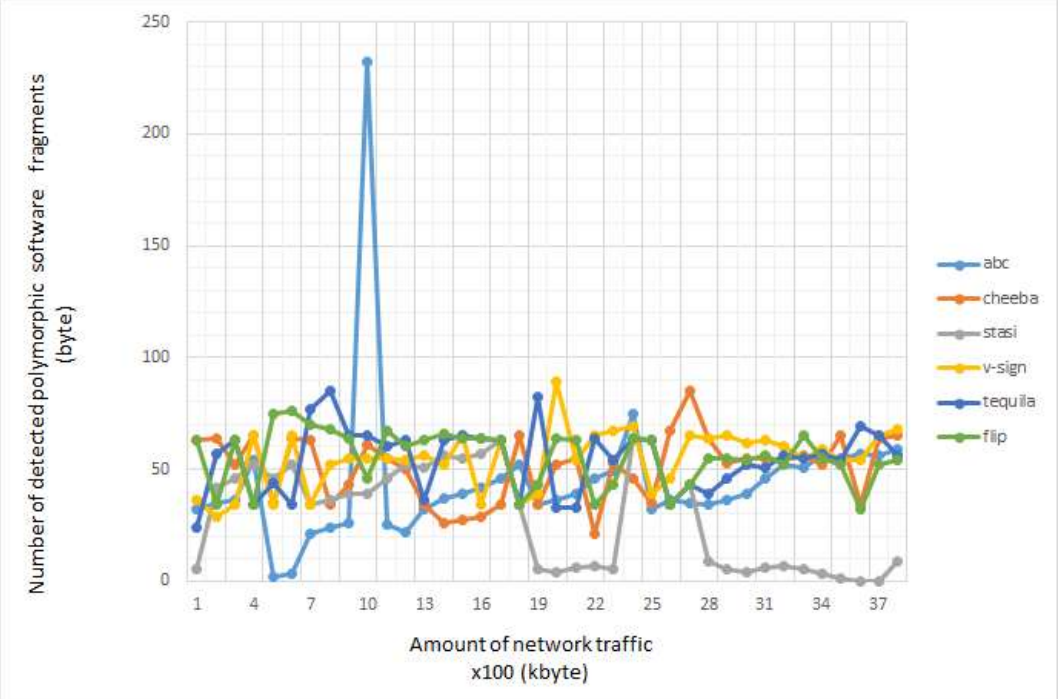


**Fig. 3 Visualization of malware detection at size «state matrices» 512x512**

The presence of at least 256 GB of RAM makes it possible to form a «state matrix» of 4096x4096 bits for training a convolutional neural network with a size of a single element of 4096 bits. In all cases, there is an increase in the degree of detection of polymorphic malware, but with an

increase in the dimension of the «state matrix», the number of false positives increases. An increase in the number of iterations leads to an increase in the accuracy of malware detection with «state matrices» up to 128x128 in size by an average of 2-3.4% with large matrices by an average of 4-4.4%. In all cases of computing, having a large amount of RAM plays an important role.

**Conclusion**

The paper considers a software model for detecting malicious polymorphic software using a convolutional neural network. Datasets for training a convolutional neural network are formed on the basis of «state matrices» of various dimensions.

At each stage of inputting data sets into the «state matrix» and the convolutional neural network, the identity of the input data was controlled by the hashing method. The Fadeev-Leverrier method was used as a mathematical apparatus for calculating the coefficients of the «state matrices» to activate the «weights» of the convolutional neural network. The developed software allows preparing data for training a convolutional neural network and detecting polymorphic malware in the presence of at least 8-10% of the source code. The calculations were carried out on a Dell Power Edge T-330 server. The source code and the full results of the research are presented in [12].

## References

1. Cheremushkin A., Cryptographic protocols: basic properties and vulnerabilities, (2009) // 272, Moscow, Academia.
2. Monappa K., Malware Analysis, (2019) //453, Moscow, DMK Press.
3. Clarence C., Freeman D., Machine Learning and Security, (2020) // 388, O`Reily.
4. Dominik Kus et al, «A false sense of security? Revisting the state of machine learning-based industrial intrusion system», (2022) // https://arxiv.org/abs/2205.09199v1
5. Jamgharyan T., «Application of a generative-adversarial network for managing a precise stochastically changeable signal sources», (2021)// Bulletin of High Technology, Stepanakert №16, pp. 49-58.
6. Burenin P., Devyanin P., Lebedenko E., Proskurin V., Tsibulya A., Security of the special-purpose operating system Astra Linux Special Edition, (2019) // 363, Moscow, Hotline Telecom.
7. Official website of Kaspersky antivirus software. Kaspersky Encyclopedia. [Online]. Available: https://encyclopedia.kaspersky.ru/knowledge/malicious-programs/
8. Jallad K., Aljnidi M., Desoki M., «Big data analysis and distributed deep learning for next-generation intrusion detection system optimization», (2022) // https://arxiv.org/abs/2209.13961
9. Zemskov A, «Construction of a matrix of transfer functions of a high-order system» (1993)// Automation and telemechanics, issue 11, pp. 40–45.
10. Official website of software for reverse engineering programs Ghidra. Software download page. [Online]. Available: https://ghidra-sre.org/
11. The official website of the comprehensive open source machine learning platform TensorFlow. [Online]. Available: https://www.tensorflow.org
12. Software source code and full research results. [Online]. Available: https://github.com/T-JN?tab=repositories

## Литература

1. Черемушкин А., Криптографические протоколы: основные свойства и уязвимости, (2009)// 272, Москва, Академия.
2. Монаппа К., Анализ вредоносных программ, (2019) //453, Москва, ДМК Пресс.
3. Clarence C., Freeman D., Machine Learning and Security, (2020) // 388, O`Reily.
4. Dominik Kus et al, «A false sense of security? Revisting the state of machine learning-based industrial intrusion system», (2022) // https://arxiv.org/abs/2205.09199v1
5. Jamgharyan T., «Application of a generative-adversarial network for managing a precise stochastically changeable signal sources», (2021) // Bulletin of High Technology, Stepanakert №16, pp. 49-58.
6. Буренин П., Девянин П., Лебеденко Е., Проскурин В., Цибуля А., Безопасность операционной системы специального назначения Astra Linux Special Edition, (2019) // 363, Москва, Горячая линия Телеком.
7. Официальный сайт антивирусного ПО Касперского. Энциклопедия Касперского. [Online].Available: https://encyclopedia.kaspersky.ru/knowledge/malicious-programs/
8. Jallad K., Aljnidi M., Desoki M., «Big data analysis and distributed deep learning for next-generation intrusion detection system optimization», (2022) // https://arxiv.org/abs/2209.13961
9. Земсков А, «Построение матрицы передаточных функций системы высокого порядка», (1993)// Автоматика и телемеханика, выпуск 11, стр. 40–45.
10. Официальный сайт программного обеспечения для реверс инженеринга программ Ghidra. Страница загрузки программного обеспечения. [Online].Available: https://ghidra-sre.org/
11. Официальный сайт комплексной платформы машинного обучения с открытым исходным кодом TensorFlow. [Online].Available: https://www.tensorflow.org
12. Исходный код программного обучения и полные результаты исследования. [Online].Available: https://github.com/T-JN?tab=repositories

## ՓԱԹՈՒՅԹԱՅԻՆ ՆԵՅՐՈՆԱՅԻՆ ՑԱՆՑՈՎ ՊՈԼԻՄՈՐՖ ՎՆԱՍԱԲԵՐ ԾՐԱԳՐԱՅԻՆ ԱՊԱՀՈՎՄԱՆ ՀԱՅՏՆԱԲԵՐՄԱՆ ՄՈԴԵԼԻ ՀԵՏԱԶՈՏՈՒՄ

**Թ.Վ. Ջամղարյան**
*Հայաստանի ազգային պոլիտեխնիկական համալսարան*

Հոդվածում ներկայացված է վնասաբեր պոլիմորֆ ծրագրային ապահովման հայտնաբերման համար փաթույթային նեյրոնային ցանցի կիրառման հետազոտության արդյունքները: Հետազոտությունն իրականացվել է abc, cheeba, december_3, stasi, otario, dm, v-sign, tequila, flip պոլիմորֆ վնասաբեր *ծրագրային ապահովման ելակետային կոդի հիման վրա կառուցած* տվյալների հավաքածուներով: Փաթույթային նեյրոնային ցանցի ուսուցումը իրականացվել է տարբեր չափերի մատրիցների միջոցով: Որպես մաթեմատիկական ապարատ օգտագործվել է Ֆադեև-Լևերիե մեթոդը: Իրականացվել է ծրագրային ապահովման իրագործման մոդելավորում տարբեր կրկնություններում և արդյունքների արտացոլում:

***Բանալի բառեր:*** մեքենայական ուսուցում, տվյալների հավաքածու, պոլիմորֆ ծրագրային ապահովում, Ֆադեև-Լևերիե մեթոդ, *ներխուժումների հայտնաբերման համակարգ:*

# ИССЛЕДОВАНИЕ МОДЕЛИ ОБНАРУЖЕНИЯ ПОЛИМОРФНОГО ВРЕДОНОСНОГО ПО СВЕРТОЧНОЙ НЕЙРОННОЙ СЕТЬЮ

**Т.В. Джамгарян**
*Армянский национальный политехнический университет*

В статье представлены результаты исследования применения сверточной нейронной сети для обнаружения вредоносного полиморфного программного обеспечения. Исследование проводилось на наборах данных сформированных на основе полиморфного программного обеспечения *abc, cheeba, december_3, stasi, otario, dm, v-sign, tequila, flip.* Формирование наборов данных для обучения сверточной нейронной сети осуществлялось с помощью «матриц состояний» различной размерности. В качестве математического аппарата использовался метод Фадеева-Леверье. Проведено моделирование работы программного обеспечения при разных итерациях и визуализация результатов.

***Ключевые слова:*** машинное обучение, полиморфное ПО, метод Фадеева-Леверье, система обнаружения вторжений, «матрица состояний».

Submitted on 07.04.2023
Sent for review on 10.04.2023
Guaranteed for printing on 19.10.2023