Application of Multifactor Analysis for the Purpose of Detecting Malicious Software Implants of the Software in Local Computer Networks

Vadym Paiuk[a], Volodymyr Kosenkov[a], Oleg Savenko[a], Andrii Nicheporuk[a], and Olena Geidarova[a]

[a] *Khmelnytskyi National University, Instytutska str., 11, Khmelnytskyi, 29016, Ukraine*

Abstract

A study of the detection of harmful implants in software. They can be of two types. In particular, in the first case, malicious implants in the software may be independent entities, and in the second case, they may be part of certain malicious software. Other cases are not considered in the work. The selected information systems of the network type, which operate in local computer networks, were selected for the study. Accordingly, the presence of harmful implants in the software is considered in local computer networks. The difficulty of detecting such harmful implants in the software lies in its ability to be in a latent state. Such a secretly included in the form of a module in the software under certain conditions can provide unauthorized access to attackers. Functionally and physically as such an object can be part of a software package. He performs the task. In addition, it can completely replace certain parts of the software. Or it can replace a certain program completely. The difficulty in detecting such malicious implants in software is that, for the most part, they make it possible to maintain the functions of useful software even when available. These functions were in the terms of reference for the project and they were required by the manufacturer. As a result, important software features are included in the system, but they are only part of it, not all.

To solve such a scientific problem, it is proposed to use the methods of multifactor analysis. Their use will indicate the presence of more important factors. This allows you to apply the results to malware implants and they may be part of some malware. The implementation of multifactor analysis methods is carried out in a network distributed malware detection system. This implementation allows you to use a ready-made system, which is tested, and add to it several methods aimed at detecting software implants. In general, this increases the reliability of detection by this system. On the other hand, it provides an opportunity to focus only on the added detection methods, as the distributed system has already been tested using other methods.

In the considered scientific problem we will use a taxonomic indicator for comparison of objects in which there is a large number of signs. That is, it can be an indicator of the presence of software implants in the software on the local network. To identify it, the application of the taxonomic method of processing statistical data of observations will allow to detect it with a certain degree of reliability. As objects of analysis, software that operates on local computer networks.

To confirm the proposed solutions, an experiment was performed with a distributed detection system, which implemented methods of multifactor analysis. The result of an experiment to identify software implants in software confirmed the viability of research and proposed solutions.

Keywords

Software implant, multifactor analysis, malicious software, local computer network, distributed detection system.

[1] IntelITSIS’2021: 2nd International Workshop on Intelligent Information Technologies and Systems of Information Security, March 24–26, 2021, Khmelnytskyi, Ukraine

EMAIL: vadympaiuk@gmail.com (V. Paiuk); vladimirkosenkov@ukr.net (V. Kosenkov); savenko_oleg_st@ukr.net (O. Savenko); andrey.nicheporuk@gmail.com (A. Nicheporuk); geydarova@ukr.net (O. Geidarova)
ORCID: 0000-0002-4104-745X (O. Savenko); 0000-0002-7230-9475 (A. Nicheporuk)

1. Introduction

The use of modern information technologies in various fields is growing. This use of information technology greatly simplifies many processes that required significant human resources. At the expense of such technologies automation of many links of technological processes of information processing is carried out. In addition, the new paradigm in the industry, which is based on the development of Industry 4.0 and the active involvement of information technology in all possible areas of human life and production, affects the positive dynamics of development. But in parallel with these processes, there are many attackers who, using weak links in the protection of such technologies, try to benefit. Therefore, the use of information technology, which does not address the protection of information flows and information, encourages malicious activity. Of particular interest to attackers are the banking sector, various financial institutions and industries. Therefore, the focus on the organization of detection of malicious actions requires primarily organizations (enterprises) that use information technology. Because they are important objects of profit from the point of view of malefactors.

Intruders access to corporate computer network resources can be software implants that are embodied in the software and hardware of computers and peripherals. They allow you to hide unauthorized access to resources. This can be done mainly through a local computer network.

The object of detection will be considered a software implant [1]. Its location will be considered local computer networks of enterprises. Software implants may not show up for a long time. This significantly complicates their detection. At the initial stage of commissioning by the software developer, checking it for the presence of software implants may be unsuccessful. That is, software implants may not be detected. This will create a problem in the future when using such software. But even if the software implant is not detected, it still has to manifest itself in a certain way, because it must communicate with certain network resources to maintain its activity and receive commands from them. If he did not communicate, then the attackers would not know where he is physically, at what addresses and so on. This activity of the software implant will lead to certain signs of its manifestation, which will change certain factors.

Software implants can store functions that are declared by the manufacturer, and they are implemented as part of the functions that are part of the software package. That is, the functions of the implant can also use the functions of a specific purpose of the software that is put into operation. This indicates the possibility of dual use of functions, which complicates the search for program implants.

Such a scientific problem is really relevant. To solve it, it is necessary to develop a mathematical apparatus and implement it in the methods of detecting software implants in software in computer networks.

Such an object can be part of a software package that performs tasks, replace completely certain parts of the software package, replace a certain required program.

2. Related works

Detection of software implants in local computer networks is carried out by various methods and means. They depend on the specific types of malware. When software implants are detected, there is a discrepancy between the results of software testing and the processes that may actually be caused by it [1]. To hide software implants, attackers have developed many approaches, algorithms, techniques and methods.

Malware-related security breaches damage users by at least $ 500 billion annually [1].

The number of malicious programs and their varieties is growing every year [2]. Attackers direct their funds to organizations and enterprises that work with information technology in local computer networks, because the financial and economic spheres motivate them the most.

There are many ways to penetrate the local networks of organizations and enterprises, some of which are described in [3-7]. The paper analyzes the strategy of an attacker to use a software implant, which is based on the use of software implants in computer software and peripherals.

The purpose of their strategy is for attackers to gain unauthorized access to system resources through the local network. The software implant in the work is considered as a secretly implemented program or software module. It is embodied in the software and as a result poses a threat to the information contained in the computer [8].

The paper considers as the object of study software implants that use the capabilities of software in local computer networks of enterprises that are malicious. There is a difficulty in identifying such a secretly operating software implant, because it does not appear during testing. Software developers embody them in useful software applications, mixing functions. But under certain conditions, such software implants can provide unauthorized access to the resources of entrepreneurs. In addition, software implants can be inactive for a long time, which makes them difficult to detect. Such software implants retain software functions. They can be implemented by some other functions that are part of the software package [9-12]. Such strategies are also implemented by well-known Trojan programs. But the object of study is such programs or program modules that are used in organizations, which are introduced by software developers when creating them for malicious use. Their hiding in the programs was carried out when handing over the finished software to the customer. Part of such a software implant in programs may be related to malicious programs such as Backdoor [9-12], and part provides other, than malicious programs such as Backdoor implementation mechanisms.

The analysis of hidden possibilities of modules in the developed software is paid attention in works [13-18].

Malicious software is a malicious tool that launches a secret sharing feature. The structure and internal algorithms of such a hidden function are given in [16]. To solve this problem, an algorithm for detecting a hidden function was developed in [17].

Models of hidden schemes of exchange of functions are developed in scientific article [18]. They are designed with a secure distributed data protocol.

In [19], the authors proposed a new method of detecting malware such as Backdoor. Artificial neural networks and object classification were used for this purpose.

The main disadvantage of this solution is a small set of proven and reliable data sets [19]. This may affect its adequacy and viability.

The model used by attackers is presented in a scientific paper [20]. It is used to hide the ways of invasion. The authors explored the possibility of using this approach to detect other types of malware, including Backdoor.

In [21], one of the methods of encoding code fragments with the help of specially designed interrupts is analyzed and substantiated, which manipulate the state of execution time when triggered and under certain conditions can continuously perform arbitrary calculations.

Problems of formalizing a terminal machine by modeling a program or system using a so-called machine with a marked final state are proposed in a scientific paper in [22]. This allows you to consider the software product as an emulator.

The complexity of this problem is analyzed using many tools in [23, 24]. For example, you can consider a hardware component, a special program, or malware. And this is a pre-necessary method of developing methods for their detection.

In works [25-27] special narrowly specialized approaches to detection of hidden software are considered. The considered methods are focused on information protection.

An active type of malware that can implant software implants are botnets [28-32]. In [28-31] the technique of detection of botnets on the basis of DNS-traffic is presented. Detect botnets based on a bot group activity property in DNS traffic that appears after a short period of time in host group DNS queries when trying to access C&C servers, migrate, run commands, or download malware updates. In [28], a method of protection against DNS evasion for detecting botnets in corporate networks is proposed.

One of the main elements that can be used to detect program implants are the functions of the application programming interface. These functions can be analyzed dynamically, as presented in [33-34]. Attackers often use the obfuscation function to mask malicious code. One of the most difficult to detect is the function of metamorphic transformation [35]. In [35] it is shown how the use of metamorphic transformations makes it possible to hide the program codes of functions.

In [36-38] the use of baits to detect abnormal and malignant manifestations was analyzed. The obtained results have shown their effectiveness and can be used in the detection of software implants.

The use of known mathematical methods for processing various events that are associated with the operation of software is presented in scientific papers in [39-47]. The considered methods can be used to identify software implants. The method of multifactor analysis for the detection of software implants in a local computer network is presented in [43]. The basis of this method is a taxonomic method of processing statistical data of observations, which is used in studies of various subject areas [44, 45].

The various methods used for the detection process require the initial stages of data preparation for processing, the purpose of which is to develop a comprehensive approach to the detection of software implants. The scientific task of detecting software implants in local networks is relevant and promising. One of the tasks that needs to be solved is to develop appropriate methods for creating effective system components for detecting software implants in local networks based on the search for abnormal manifestations, taking into account multifactor analysis.

3. The Approach to Application of Multifactor Analysis for the Purpose of Detecting Malicious Software Implants in Local Computer Networks

In our case, the objects are computer networks, and the signs may be [30, 31]: the presence of software modules that do not meet the purpose of the process; the presence of files related to operating systems, and which form open processes that do not meet the purpose of the process; there is a high intensity for I / O operations from a certain process, and so on. (11 signs are given, although their number may be higher).

The basis for research is a matrix of observations X:

$$X = \begin{bmatrix} x_{11} & x_{21} & \dots & x_{1k} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & x_{2k} & \dots & x_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ x_{i1} & x_{i2} & \dots & x_{ik} & \dots & x_{in} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ x_{\omega 1} & x_{\varpi 1} & \dots & x_{\omega k} & \dots & x_{\omega n} \end{bmatrix} \tag{1}$$

where x_{ik} is the number of manifestations of the *k*-th feature in the *i*-th object during the observation period; n is the number of features; ω is the number of objects.
How isotonic and isomorphic (structural) ordering of objects, Chekanovsky's method for research of subsets on homogeneity is resulted.

In economic research, a taxonomic indicator of the level of development is used to compare objects that are characterized by a large number of features [43]. In our case, this may be an indicator of the presence of software implants in the local network. And this will be a further development of research, which is presented in [1].

The first stage is the standardization of features in the matrix (1) and its transformation into a matrix Z.

- the expert constructs a square matrix of $n \times n$ pairwise comparisons of features, which has the property of inverse symmetry $a_{ji} = {}^{1}\!/_{a_{ij}}$;
- eigenvectors of priorities are calculated, for which all elements of a row are multiplied and the root of the *n*-th degree is taken from result, and then the received number is divided by the sum of such numbers of a column and estimates of a vector of priorities ($x_1, x_2, \dots, x_n$) are received;
- the matrix $n \times n$ is multiplied by the column of the priority vector and a column with ($y_1, y_2, \dots, y_n$) is obtained, which shows the degree of importance of each feature.

Then, before the transition from the matrix X to the matrix Z, the results of observations in the matrix (1) must be multiplied by the coefficients y_k , respectively.

In [42], the transition to the matrix Z is proposed in the following sequence:

$$Z_{ik} = \frac{X_{ik} - \bar{X}_k}{S_k}, \tag{2}$$

where

$$\bar{X}_k = \frac{1}{\omega}\sum_{i=1}^{\omega} X_{ik}, \tag{3}$$

$$S_k = \left[\frac{1}{\omega}\sum_{i=1}^{\omega}(X_{ik} - \bar{X}_k)^2\right]^{\frac{1}{2}}. \tag{4}$$

$k = 1,2\ldots,n$ the value of the sign *k* for the unit *i*; $\overline{X}_k$ is the arithmetic mean of the sign *k*; S_k - standard deviation of the sign k; Z_{ik} is the standardized value of the characteristic *k* for the unit *i*.

Next, the so-called development standard is formed, which represents a point with $Z_{01}, Z_{02}, \ldots, Z_{0n}$ coordinates. These coordinates represent the standards or valid values of the features.

Then the distance between the points-units of the matrix *Z* and the point is determined by the formula:

$$\text{C}_{i0} = \left[\sum_{k=1}^{n}\left(Z_{ik} - Z_{0k}\right)^2\right]^{\frac{1}{2}} \quad (i = 1, \ldots, \omega). \tag{5}$$

At these distances, the indicator of the presence of software implants is calculated:

$$d_i^* = \frac{C_{i0}}{C_0}, \tag{6}$$

where

$$C_0 = \bar{C}_0 + 2S_0, \tag{7}$$

$$\bar{C}_0 = \frac{1}{\omega}\sum_{i=1}^{\omega} C_{i0}, \tag{8}$$

$$S_0 = \left[\frac{1}{\omega}\sum_{k=1}^{n}(C_{i0} - Z_{0k})^2\right].^{\frac{1}{2}} \tag{9}$$

The indicator d_i^* can be in the range 0… .1. The closer this value is to zero, the more likely it is that there will be no software implants in the facility.

The indicator is used to statically characterize a set of objects. For a more in-depth analysis, you need to consider the dynamic characteristics of a single object and then a set of objects.

Then, based on the results of observations for several periods of time, a matrix of observations *X* is formed for one object:

$$X = \begin{bmatrix} x_{11} & x_{21} & \ldots & x_{1k} & \ldots & x_{1n} \\ x_{21} & x_{22} & \ldots & x_{2k} & \ldots & x_{2n} \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ x_{i1} & x_{i2} & \ldots & x_{ik} & \ldots & x_{in} \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ x_{t1} & x_{t2} & \ldots & x_{tk} & \ldots & x_{tn} \end{bmatrix} \tag{10}$$

where x_{ik} - the value of the sign *k* in the period *i*.

Then, as shown above, there is a process of standardization (matrix *Z*), the standard is built P_0 .

The taxonomic index d_i^* is determined by formula (6), where

$$\mathrm{C}_{i0} = \left[\sum_{k=1}^{n}(Z_{ik} - Z_{0k})^2\right]^{\frac{1}{2}} \quad (i = 1, \dots, t), \tag{11}$$

$$C_0 = \bar{C}_0 + 2S_0, \tag{12}$$

$$\bar{C}_0 = \frac{1}{t}\sum_{i=1}^{t} C_{i0}, \tag{13}$$

$$S_0 = \left[\frac{1}{t}\sum_{k=1}^{t}(C_{i0} - \mathrm{C}_0)^2\right]^{\frac{1}{2}}. \tag{14}$$

So, now the indicator d_i^* describes the dynamics of changes in the sets under study, but for one object.

You can now proceed to the dynamic characterization of a set of objects. If you denote the observation matrix of an object j by a symbol X_j, the aggregate matrix for objects will remain as a block matrix:

$$X_0 = \left[X_1, Z_2, \dots, Z_{\omega n}\right], \tag{15}$$

Given that all objects in the network are of the same type - they are computers, the standard P_0 may remain from the previous analysis.

The generalized index d_i^* is determined by formula (6), where:

$$\mathrm{C}_{i0} = \left[\sum_{i=1}^{\omega}\sum_{k=1}^{n}(Z_{ik} - Z_{0k})^2\right]^{\frac{1}{2}} \quad (i = 1, \dots, t). \tag{16}$$

C_0, S_0 are defined by formulas (12), (13), (14).

Consider the variable n as the number of features; ω - number of objects; Z_{ik} - standardized value of the sign *k* in the period *t*.

The calculated value d_i^* describes the process with the dynamic characteristics of all objects. But according to [44-46], the directions of changes of individual components by the total value of the indicator are not taken into account here. Therefore, in [42] it is proposed to replace the distance C_{i0} with $C_{(i0,j)}$:

$$\mathrm{C}_{i0,j} = \left[\sum_{k=1}^{n}\left(Z_{ik} - Z_{0k,j}\right)^2\right]^{.\frac{1}{2}} \tag{17}$$

where $i = 1, 2, \dots, t$: $j = 1, 2, \dots, \omega$; $Z_{(0k,j)}$ - coordinates of the development standard of the object *j*.

Then the dependence (16) for the square of the distance for the aggregate indicator and the square of the distance for the individual indicators can be written as:

$$C_{i0}^2 = \sum_{j=1}^{\omega} C_{i0,j}^2 . \tag{18}$$

Taking into account $C_{i0} = C_0 \cdot d_i^*$ and $C_{(i0,j)} = C_{(0,j)} \cdot d_{ij}$ the dependence between aggregate and individual indicators will remain:

$$d_i^* = \frac{\sum_{j=1}^{\omega} C_{0,j}^2 \cdot d_{i,j}^*}{C_0^2}. \tag{19}$$

The obtained formula allows to estimate the influence of individual taxonomic indicators of objects d_{ij}^* on the general taxonomic indicator of the set of objects d_i^*. Here, as before, this figure is in the range (0, 1).

The result of the indicator refers to one of the five intervals that allow us to assess the level of possible presence of software implants.

4. Experiments and evaluation

For experiments, a distributed system [4, 30, 48-52] was used to detect malware, which will include the ability to detect software implants based on implemented methods. An appropriate method was implemented to check the efficiency of the classifier in the structure of the distributed system. The result of the determination was the dependence of the percentage of detected botnets containing software implants. Experimental studies were performed for the classifier without adding copies of the created botnets, ie the test was performed without training the classifier on the created samples. Botnets that use a strategy to gain complete control by activating their components were selected for the experiment. That is, software implants were present at every computer station. The results of the calculation of various indicators are presented in table 1. The experiments involved determining the following metrics for the detection of bot nodes: P_1– percentage of harmful vectors belonging to a certain class; P_2 – the percentage of vectors of harmful actions belonging to this subclass of the class in relation to all test vectors; P_3 – the percentage of correctly detected botnet nodes; P_4 – the percentage of incorrectly classified botnet nodes as benign applications; P_5 – the percentage of incorrectly assigned bot nodes to one of the botnet classes.

Table 1.
Results of Experiments

Metrics – Class:	C0	C1	C2	C3	C4	C5	C6	Mean
P_1, %	89,74	83,29	72,66	86,30	92,04	92,18	96,60	88,44
P_2, %	85,60	83,37	72,38	85,19	98,68	93,72	96,40	88,22
P_3, %	92,21	84,31	72,03	89,57	90,63	88,52	93,78	87,82
P_4, %	7,79	14,37	27,97	10,43	7,27	11,48	6,22	11,60
P_5, %	0	1,02	0	0	1,15	0	0	0,31

The result is approximately 26.7% of the total number detected. The intensity of manifestations of software implants is much lower than the typical manifestations of botnets. Thus, software implants used by botnets can be detected by distributed systems [30].

5. Discussion and Future Work

Thus, software implants create problems for software users. This is especially true for organizations and businesses. The advantage of attackers who use software implants is that they use hidden software functions. The difficulty in detecting such program a implants is that the processes occur in local networks. Attackers develop and use such tools in various malicious models.

6. Conclusions

On-premises software implants in software create problems for PC users. The proposed method for detecting software implants allows you to assess the degree of availability in the software. The application of the proposed solution in a distributed malware detection system has increased the efficiency of software implant detection by 4% through the use of multidimensional analysis to detect software implants in software on a local computer network.

The direction of further research will be the development of methods for detecting software implants based on their behavioral signatures.

7. References

[1] G. Sanjam, C. Gentr, S. Halevi, M. Raykova, A. Sahai, B. Waters, Hiding Secrets in Software: A Cryptographic Approach to Program Obfuscation. Communications of the ACM, 59 5 (2016) 113-120

[2] McAfee Mobile Threat Report Q1, 2019. URL: https://www.mcafee.com/enterprise/en-us/assets/reports/rp-mobile-threat-report-2019.pdf

[3] K. Drozd, O. Zashcholkin, O. Martynyuk, J. Ivanova, J. Drozd, Development of Checkability in FPGA Components of Safety-Related Systems, CEUR WS 2762 (2020) 30-42

[4] S. Lysenko, K. Bobrovnikova, S. Matiukh, I. Hurman, O. Savenko, Detection of the botnets' low-rate DDoS attacks based on self-similarity, International Journal of Electrical and Computer Engineering 10 4 (2020) 3651-3659

[5] B. Anderson, D. Quist, J. Neil, C. Storlie, T. Lane, Graph-based malware detection using dynamic analysis. Journal in Computer Virology, 7 (2011) 247-258

[6] N. Runwal, R. M. Low, M. Stamp, Opcode Graph Similarity and Metamorphic Detection. Journal in Computer Virology, 8 (2012) 37-52

[7] A. Nagaraju Metamorphic malware detection using base malware identification approach. Journal Security and Communication Networks, 7 (2014) 1719-1733

[8] DSTU 3396.2-97 Protection of information. Technical protection of information. Terms and definitions. State Committee of Ukraine, Kyiv (1997) [in Ukrainian].

[9] B. Chen, W. Carvalho, N. Baracaldo, H. Ludwig, B. Edwards, et al, Detecting Backdoor Attacks on Deep Neural Networks by Activation Clustering. CEUR WS 2301 (2019)

[10] Adups Backdoor. URL: https://www.kryptowire.com/adups_security_analysis.html.

[11] K. Alminshid, M. N. Omar, Detecting backdoor using stepping stone detection approach, in: Proceedings of 2013 Second International Conference on Informatics & Applications (ICIA), Lodz, Poland, 2013, pp. 87-92

[12] J. Zaddach, A. Kurmus, D. Balzarotti, E.-O. Blass, A. Francillon, et al., Implementation and Implications of a Stealth Hard-drive Backdoor, in. Proceedings. of 29th Annual Computer Security Applications Conference, New Orleans, Louisiana, US, 2013.

[13] T. F. Dullien, Weird machines, exploitability, and provable unexploitability, IEEE Transactions on Emerging Topics in Computing, 99 (2017) 1-15

[14] S. L. Thomas, T. Chothia, F. D. Garcia, HumIDIFy: A Tool for Hidden Functionality Detection in Firmware, in: Proceedings of 14th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, Bonn, Germany, 2017, pp. 279-300

[15] S. L. Thomas, T. Chothia, F. D. Garcia, Measuring the Importance of Static Data Comparisons to Detect Backdoors and Undocumented Functionality, in: Proceedings of 22nd European Symposium on Research in Computer Security. Oslo, Norway, 2017, pp. 513-531

[16] A. Schönegge, The Hidden Function Question Revisited, in: Proceedings of Algebraic Methodology and Software Technology: 6th International Conference, AMAST '97. Sydney, Australia, 1997, pp. 451-464

[17] The Secret Code of Software Validation….In 5 Easy Steps. URL: https://www.cebos.com/blog/the-secret-code-of-software-validation-in-5-easy-steps/

[18] Y. Kawamoto, H. Yamamoto, Secret function sharing schernes and their applications to the oblivious transfer, in: Proceedings of IEEE International Symposium on Information Theory, 2003, pp. 281-295

[19] B. Chen, W. Carvalho, N. Baracaldo, H. Ludwig, B. Edwards, et al, Detecting Backdoor Attacks on Deep Neural Networks by Activation Clustering, CEUR Workshop 2301 (2019).

[20] J. Tarhio, E. Ukkonen, Approximate Boyer Moore String Matching. SIAM Journal on Computing 22 2, (1993) 243-260

[21] Y. Kondratenko, N. Kondratenko, Soft Computing Analytic Models for Increasing Efficiency of Fuzzy Information Processing in Decision Support Systems. Chapter in book: Decision Making: Processes, Behavioral Influences and Role in Business Management, R. Hudson (Ed.), Nova Science Publishers, New York, 2015, 41-78

[22] V. Proskurin, Software malicious implant in secure systems. URL: http://www.crime-research.ru/library/progwir98.htm [in Ukrainian].

[23] O. V. Kaarin, Program protection theory and practice. MGUL, 2004 [in Russian].

[24] O. V. Kaarin, Computer system software security. MGUL, 2003 [in Russian].

[25] V. F. Shanugin, Protection of computer information. Effective methods and tools: a textbook. DMK Press, 2008 [in Russian].

[26] V. F. Shanugin, Protection of information in computer systems and networks. DMK Press, (2012) [in Russian].

[27] G. Balakrishnan, T. Reps, WYSINWYX: What You See Is Not What You eXecute. in: Proceedings of ACM Transactions on Programming Languages and Systems, Vol. 32, Issue 6, 2010.

[28] S. Lysenko, K. Bobrovnikova, O. Savenko. A Botnet Detection Approach Based on The Clonal Selection Algorithm, in: Proceedings of 2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies, DeSSerT-2018, Kyiv, Ukraine, 2018, pp. 424-428.

[29] S. Lysenko, O. Pomorova, O. Savenko, A. Kryshchuk and K. Bobrovnikova DNS-based Anti-evasion Technique for Botnets Detection, in: Proceedings of the 8-th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, Warsaw, 2015, pp. 453–458.

[30] S. Lysenko, K. Bobrovnikova, O. Savenko and A. Kryshchuk, BotGRABBER: SVM-Based Self-Adaptive System for the Network Resilience Against the Botnets' Cyberattacks, Communications in Computer and Information Science, 1039 (2019) 127-143. doi: 10.1007/978-3-030-21952-9_10

[31] O. Savenko, S. Lysenko, A. Kryschuk, Multi-agent based approach of botnet detection in computer systems Communications in Computer and Information Science, 291 (2012) 171-180

[32] S. Taheri, A.M. Bagirov, I. Gondal, S. Brown. Cyberattack triage using incremental clustering for intrusion detection system Internation Journal of Information Security, 19 (2020) 597–607. doi: https://doi.org/10.1007/s10207-019-00478-3.

[33] O. Savenko, A. Nicheporuk, I. Hurman, S. Lysenko, Dynamic signature-based malware detection technique based on API call tracing CEUR-WS 2393 (2019) 633-643

[34] A. Drozd, J. Drozd, S. Antoshchuk, V. Nikul, M. Al-dhabi, Objects and Methods of On-Line Testing: Main Requirements and Perspectives of Development, in: Proceedings of IEEE East-West Design & Test Symposium, Yerevan, Armenia, 2016, pp. 72 – 76. doi: 10.1109/EWDTS.2016.7807750

[35] O. Pomorova, O. Savenko, S. Lysenko, A. Nicheporuk Metamorphic Viruses Detection Technique based on the Modified Emulators, CEUR-WS 1614 (2016) 375-383

[36] T. Sochor, M. Zuzcak, Study of Internet Threats and Attach Methods Using Honeypots and Honeynets, Computer Network 431 (2014 118–127

[37] T. Sochor, M. Zuzcak, Attractiveness Study of Honeypots and Honeynets in Internet Threat Detection, Computer Networks 522 (2015) 69-81. doi: 10.1007/978-3-319-19419-6 7.

[38] P. Owezarski, A near real-time algorithm for autonomous identification and characterization of honeypot attacks, in: Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security, ser. ASIA CCS ’15. New York, NY, USA: ACM, 2015, pp. 531–542.

[39] J. Mazel, P. Casas, P. Owezarski. Sub-Space Clustering and Evidence Accumulation for Unsupervised Network Anomaly Detection, in: Proceedings of the Third International Conference on Traffic Monitoring and Analysis, ser. TMA’11. Berlin: Springer-Verlag, 2011, pp. 15–28.

[40] N. A. Rosli, W. Yassin, M.F. Faizal, S.R. Selamat Clustering Analysis for Malware Behavior Detection using Registry Data. (IJACSA) International Journal of Advanced Computer Science and Applications 10 12 (2019) 93-102.

[41] S. Bezobrazov, A. Sachenko, M. Komar, V. Rubanau, The method of artificial intelligence for malicious applications detection in android OS, International Journal of Computing, 15(3) (2016) 184-190

[42] M. Kolisnyk, V. Kharchenko, I. Piskachova, Research of the attacks spread model on the smart office’s router, International Journal of Computing, 19(4) (2020) 629-637.

[43] V. Pluta Comparative multidimensional analysis in economic research: methods of taxonomy and factor analysis, Statistics, 1980 [in Russian]

[44] B. N. Igumnov, T. P. Zavgorodnyaya, Cybernetic bases of construction of economic systems for the enterprises, TUP, 2000 [in Russian]

[45] G. Saat, K. Kerno, Analytical planning. Organization of systems: Translated from English, Radio and communication, 1991 [in Russian]

[46] A. V. Andreychikov, O. N. Andreychikova Analysis, synthesis, planning solutions in economics, Finance and Statistics, 2001 [in Russian]

[47] A. Melnyk, V. Melnyk, Remote Synthesis of Computer Devices for FPGA-Based IoT Nodes, in: Proceedings of 2020 10th International Conference on Advanced Computer Information Technologies, ACIT 2020, pp. 254-259

[48] A. Drozd, M. Lobachev, J. Drozd, “The problem of on-line testing methods in approximate data processing,” Proc. 12th IEEE International On-Line Testing Symposium, Como, Italy, pp. 251–256, 2006. DOI: 10.1109/IOLTS.2006.61.\

[49] J. Drozd, A. Drozd, M. Al-dhabi, “A resource approach to on-line testing of computing circuits,” Proc. IEEE East-West Design & Test Symposium, Batumi, Georgia, 2015, pp. 276 – 281. DOI: 10.1109/EWDTS.2015.7493122

[50] Lysenko, S., Savenko, O., Bobrovnikova, K., Kryshchuk, A., Savenko, B.: Information Technology for Botnets Detection Based on Their Behaviour in the Corporate Area Network. In: International Conference on Computer Networks, 2017, pp. 166-181. Springer, Cham.

[51] Lysenko, S., Savenko, O., Bobrovnikova, K., Kryshchuk, A.: Self-adaptive System for the Corporate Area Network Resilience in the Presence of Botnet Cyberattacks. In: International Conference on Computer Networks, pp. 385-401. Springer, Cham (2018).

[52] Savenko O., Lysenko S., Nicheporuk A., Savenko B. Approach for the Unknown Metamorphic Virus Detection. The 9-th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications : Proceedings (Bucharest, Romania, September 21–23, 2017). Bucharest, 2017. Vol. 1. Pp. 453–458.