INTEROPERABILITY OF DISTRIBUTED MULTIPLE SYSTEM FOR MALWARE DETECTION BASED ON COMPONENTS LEVELS OF SAFETY

UDC 004.491

DOI: https://doi.org/10.35546/2313-0687.2018.24.78-92

SAVENKO Oleh

PhD, Professor, Dean of the Faculty of Programming, Computer and Telecommunication Systems, Khmelnytsky National University, Khmelnytsky, Ukraine, E-mail: savenko_oleg_st@ukr.net, 0000-0002-4104-745X

Abstract. The work out the developed method of interaction of components of distributed multi-level system of detection of malicious software on the basis of decentralized and self-organized architecture in local networks. Its feature is the synthesis of its requirements of distribution, decentralization, multilevel and self-organization. This allows you to use it autonomously. The basis of the distributed distributed system is its structural components, which are represented by autonomous software modules that can be in different states. The transition between module states is based on a defined set of transitions. Interaction and communication between autonomous software modules is based on their presence in certain states during operation and is determined by the rules of the developed method. Distributed system is a responsive system that will monitor selected events. Each program module places a resident mechanism, the motive mechanisms for the transition between states, the transitions between which are given by subsets of transitions, the data for which will be formed using the technologies of artificial intelligence. In addition, the feature of the components of the system is the same organization, which allows the exchange of knowledge in the middle of the system, which, unlike the known systems, allows us to use the knowledge gained by separate parts of our system in other parts. The developed system allows to fill it with subsystems of detection of various types of malicious software in local area networks. The method of interaction of components of a distributed multilevel detection system of malicious software provides a procedure for communication between parts of the system and the exchange of knowledge between them. It will be used to organize the interaction of system components and maintain its integrity. In order to solve the problem of the direct detection of malicious software in local area networks, methods will be applied that will be applied to the lower level of the system, which will include the architectural features of the distributed system and the technology of detecting the malicious software-based software. However, the developed method of interaction includes the ability to determine the state of a distributed multi-level system, depending on the states of individual modules, and on its basis, in accordance with it will be decided on the further operation of the system as a whole and its configuration. The method regulates the actions of the part of the system that relates to the bundling software of the distributed system. The conducted experiments on the use of the developed distributed system showed the possibility of attracting to the detection of the malicious software of computing power of

other hosts of the local network. The obtained results of experiments show an increase in the reliability of the detection of malicious software.

Keywords*: malware, distributed multi-level system, decentralized system, computer systems, local area network.*

Formulation of the problem. The growth of the number of computer systems and the spread of information technology in various industries and spheres, their integration into the global Internet network, as well as the growing opportunities for obtaining financial returns that appear at the same time, motivate malware developers to increase and spread them [1, 2]. Trends in the development of technology for the creation and spread of malware-many of the security demonstrate an active expansion of the technical capabilities of such tools. Modern malware is a complex, multi-functional software system and complex that is built using effective methods to create software and malicious code spreading methods.

Distribution of malware in the information systems of lo cial networks creates problems for users. Available means of its appearance today do not meet the needs of users. This is especially true of the task of detecting malware prior to it, at the stage of its direct distribution. As a rule, detection of malware comes from already after it was spread over a period of time and was carrying out destructive actions. A variety of antivirus tools that detect malicious software at different stages of its operation are known to prevent its full detection [1]. A special place is occupied by antivirus products [3-7] that detect malware in local area networks. They allow you to take advantage of an organization with more computing power than individual computer systems. They are mainly used in corporative networks of organizations and enterprises. Such network detection systems have a centralized architecture used by malicious people to block them, after detecting and blocking their centers.

Distribution of malware creates problems for users of computer systems. Existing detection systems do not provide full detection. Therefore, the problem of developing new methods and systems for detecting malicious software is relevant.

Analysis of recent research and publications. For network detection systems, methods have been developed, which are possible mainly on the server or on corporate or local networks. Most of these techniques are developed using technologies and components of artificial intelligence. As a rule, modern malware detection systems contain sets of many methods and their combinations. This is influenced by the growth of malicious software varieties. Let's consider more well-known systems and methods for detecting malicious software.

The known implementation of such detection systems mainly has a single control center with a certain level of centralization. These include: ESET Endpoint Security for Windows Endpoint Protection in corporate networks [1], Dr.Web CureNet! [2], Symantec Endpoint Protection [3], Malwarebytes Endpoint Security [5], "Cisco® Network Admission Control (NAC)»[6]. Kaspersky Administration Kit [6] has implemented the principle of autonomous decision-making in the network-based detection system. But in the future, the administrator controls the decisions taken. This indicates the presence of partial centralization in the architecture of this network detection system.

Known malware detection tools are built on top-notch methods that do not sufficiently take into account all the stages of operation and possible structures, which reduces the authenticity of the detection.

The authors of the paper [8] proposed a system of identification and classification for megye cyber attacks. To implement the system, a combination of different methods of artificial machine learning, namely neural networks, the immune system, neuro-physical classifiers and the method of reference vectors, is proposed. A distinctive feature of the proposed system is the multi-level analysis of network traffic, which makes it possible to detect signature attacks and combine a set of adaptive detectors based on machine learning techniques.

The system for detecting cyberattacks on the basis of the involvement of neural network immune detectors is presented in [9]. The decision on the possible impact of the malicious software is carried out with the involvement of a system of neural network detectors based on the algorithm of Mamdani.

The system of detection based on the selection of characteristic features of the flow of the program is presented in [10]. The proposed system involves building a

graph of the flow of malware control, and then converting it into a vector pro-stripe.

In [11], the authors identify a coordinated form of organized cyber attacks in botnet components, in which they conduct synchronized attacks in the form of groups. The similarity of the activities of cooperative groups is used as an effective measure for distinguishing bots from ordinary users. In this paper an approach is proposed for the analysis of behavior based on the histogram. To determine the number of web requests and their diversity over time using HTTP bots. As a result, the detection method is based on the correlation analysis of communication histograms designed to detect HTTP Botnets based on the similarity and correlation of their group activity.

The development of methods without taking into account the fundamental features of bot network architectures, allows attackers to bypass the tools that use typical representations. Sys-theme of modeling agents of various architectures of botnets, taking into account the various mechanisms of their functioning, is presented in [12]. It is based on the need to take into account the specialties of construction and structure. This is important for gathering the characteristics of botnets.

The botnet detection system presented in [12] is based on the analysis of traffic, and tac-toe modifications of the methods are oriented on comparing the results of the analysis of traffic with templates of the base of anomalies. The disadvantage is the need for a constant analysis of traffic and the allocation of important characteristics that may vary by intruders. This does not take into account the architecture of the botnet and packet blocking in the future does not guarantee their repetition.

In [13-14], detection methods are based on signatures. The results are applicable to known bots. They provide for control of each package and compare them with pre-configured signatures and attack patterns in the database. The common disadvantage of these methods is the need to update templates, which affects the system's failure to display new botnets or their nodes.

The analysis showed that for detecting malware, known systems carry out analysis of network traffic, audit files, packets transmitted over the network, checking the configuration of open network services. To establish the fact of a violation of the work of the systems, various methods of motor training are used, namely, neural networks, artificial immune systems, the method of reference vectors, Bayesian networks, and fuzzy clustering [8-14]. The main disadvantage of known systems is their host-oriented approach to detecting malicious software, and for network detection systems, the presence of centralization in decision-making or maintenance.

Therefore, further development of the theory and practice of creating distributed systems for detecting malware is a topical scientific and technical problem.

The aim of the study. In order to effectively apply malware detection methods and means, it is necessary to develop a system that includes a sufficient number of implemented effective methods in the form of appropriate subsystems, has a potential for upgrading and takes into account future development trends as antivirus agents, and malware.

The purpose of the work is to develop the theory and practice of creating distributed multilevel detection systems to increase the reliability of malware detection in computer systems of local networks based on decentralization and self-management. Multilevelness involves the inclusion in the system of different detection methods and their placement at different levels. Self-organization serves as the basis for the functioning of a rolled-up system with decentralized architecture.

The need for distributed multi-level detection systems for malware-malware detection in local computer systems is needed to attract other local area network hosts for the detection process. This will increase the reliability of its detection.

Presentation of research material. In order to increase the authenticity of the detection of malicious software, it is proposed to use decentralization of the system in distributed systems in local networks and to apply the developed method of interaction of the components of the distributed detection system of the malicious software to coordinate the components of the system. It establishes the order of communication between the components of the system and the exchange of knowledge between them based on the security levels of the system. They are dynamically determined at certain moments of time. It will be used to solve top-level tasks of the interaction organization. Only for the organization of the interaction of components of the system and the presentation of its integrity. To solve the problem of

directly detecting malicious software on local networks, methods will be applied that will apply to the lower level of the system. They will include the architectural features of the distributed system and the technology of detecting malicious software. The generalized scheme of the main components of the distributed system in local computer networks is shown in Fig. 1/ Architecture of the distributed multilevel detection system of the malicious software is presented in [16].

The architecture of a distributed multi-level system based on decentralization and self-organization to detect malware. Taking into account that the malware detection process will be conducted on local networks, the choice of the model of the system's operation should involve the inclusion of information from all computer systems of the local network, that is, placement in all computer systems of the system. This is necessary to increase the efficiency and reliability of detection by taking into account information on the state of other computer systems for decision-making in a particular computer system. These basic requirements that a system should be placed on the network in each computer system, affect the choice of model of its architecture. Also important for such systems is that the center of decision-making of the system is not presented and identified unambiguously, since its detection will lead to an attack on it to remove the entire system from the working state. The system should be constructed so that its components in the computer systems of the local network communicate effectively with each other for the exchange of information about the state of the computer systems in order to provide additional information for decision-making. In addition, the malware detection system must be structured accordingly in order to be able to grow and increase it should not slow down the detection process.

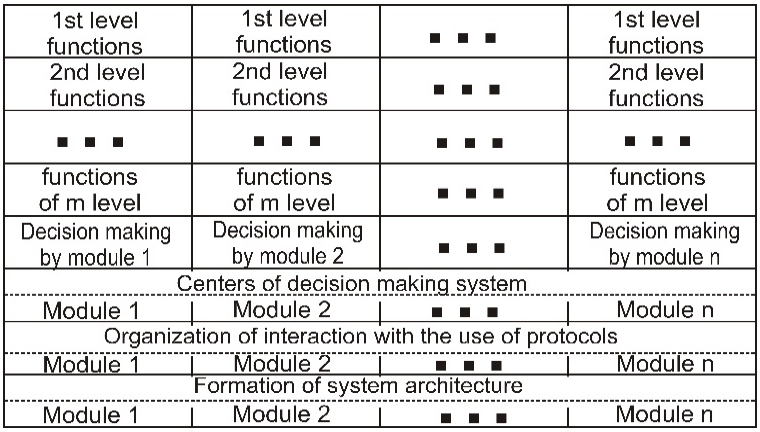

Fig.1 The generalized scheme of the main components of the distributed system

Structural components of the built-in distributed system appear as autonomous software modules. They can be in different states. The transition between module states is based on a defined set of transitions. Interaction between standalone software modules. It is based on their presence in certain states during operation. Distributed system is a responsive system that will monitor certain events. Each program module contains a resident mechanism, moving mechanisms for the transition between states.

Since the detection system is distributed, multilevel, decentralized and self-organized, then it is necessary to establish rules for its functioning and its components.

Method of interaction of components of distributed multi-level system of detection of malicious software on the basis of decentralization and self-organization. The method of component interaction supports the integrity of the system, changes its configurations, establishes the order of communication between the components of the system and the exchange of knowledge between them. It will be used to solve the tasks of organizing the interaction of components of the system. To solve the problem of the direct detection of malware, software will apply methods that are related to the lower level of the system, which will include the

architectural peculiarities of the distributed system and the technology of detecting the malicious software.

Initial conditions for work The distributed system according to the method of interaction of its components are as follows: the launch of the software module in a specific host is successful but successful; no less than in the two systems of the network, the start function was successful; programm module has already been preinstalled successfully. Then, the following protocols that occur on the network that are related to the functioning of the system, will be presented by some steps:

1) determining the states of program modules;

2) processing responses from the software module to the sent packages;

3) software uncertainty processing associated with the absence of responses to sent packages;

4) scanning a given port computer systems;

5) assessment of the status of the software module and its verification between the rest of the software module distributed multi-level system at the stage of the exchange of messages;

6) determining the state of a decentralized distributed system;

7) making a decision on the further work. distributed multi-level system in general on the basis of studying its state by software modules;

8) the removal of the active software module from the distributed multi-level system as a result of switching off the software module;

9) events that activate methods for detecting malicious software, affect the state of the state of the software module distributed by the distributed multi-level system; to carry out research of other computer systems for the presence of similar activities and exchange of received results;

10) processing and optimization of statistical data accumulated in the system by each module separately;

11) knowledge sharing in the middle of a decentralized distributed system;

12) joint execution of tasks with components distributed multi-level system;

13) the work is distributed by the basis system, consisting of only one component of system;

14) additional work of the distributed multi-level system with new software module.

Determination of the states of program modules distributed multi-level system. Determination of the status of each software module in the computer systems , in which the distributed multi-level system at the stages of the start of the software module during its initial installation, with daily loading of the computer systems, during the functioning of the computer systems and at the completion of the computer systems. Determination of the status of the program module in the computer systems includes verification of the computer systems, its software and directly the level of activity of the software module itself. Inclusion of information in the internal bases of each software module of the system. Comparing the results of scanning the computer systems with the previous scan results for a certain period stored in the scan database. If the scan results do not match the previous, then the computer systems is blocked and a corresponding message is displayed. If the basic scan settings match, then the software module continues to work. Preparation and formation of the message about your condition by each module of the system. Entering a package message into the software module (sent and received). Sending message packets about your condition to all other parts. distributed multi-level system in accordance with the specified registry of software modules of the system and computer systems in which they are located. Saving information about events by the numbers and addresses of program modules distributed multi-level system, which are included in the database of software modules of the remaining computer systems during installation.

Processing responses from the software modules computer systems to the sent packages. Receive responses to the sent message about the status of the software module. If the submitted package was delivered successfully, then the corresponding response is sent, and it will necessarily be received separately after the first packet sent from all components of the system. Expecting a response by the program module of the system occurs at a given time interval, which is calculated at the initial installation of the system and takes into account the technical capabilities of the network at the speed of packet transmission, and also certain criteria can be entered, the execution of which indicates the need to wait for the message on the sent package, rather than switching to next step. Taking into account the disabled computer systems

that contain the software modules distributed multi-level system and in which the software modules are not active, to assess the state of integrity the system is distributed and its structure at certain moments of time. Receiving responses from everyone registered in the distributed multi-level system, a software module from the rest of the computer systems on the successful receipt of their package with a statement of their status from all software modules distributed multi-level system. Conduct analysis of received answers from the software modules of the remaining computer systems and analysis from which the computer systems received replies, but not received. Processing of the response to the successful receipt of the package with the message about the status of the program module, which sent this message to all the rest of the software modules with distributed multi-level system, from a certain number of software modules and determination of those from which no response was received. Processing of the event, which consists in not receiving the response from anysoftware modules in the established time requirements. Adding the received information to the module's message base. Formation or confirmation of integrity distributed multi-level system with active program modules. For each package, which is formed after the computer systems is turned on and sent to the rest of the computer systems, it is mandatory to receive the application and to include the software modules in the register of the active software modules distributed multi-level system.

Processing by the program module of uncertainties associated with the lack of responses to sent packages. If the response to the sent packet from a given software module of a particular computer systems is not received within a specified time interval or the definition of such a fact by other criteria, then scan the required port of a given computer system. If the answer is received that the packet was not delivered due to crashes in the transfer system, then retry sending the packet to the specified computer systems.

Scan a given port computer systems . The port number through which the exchange of messages between program modules is defined during configuration when the software module is installed in the computer systems. Additionally, several more ports can be installed as backup to improve system durability. If the port scan is successful, that is, the port is accessible, then the mark in the database of sent and received messages is made and the software modules for this computer systems goes into the packet waiting state from it. Otherwise, that is, if the scan is unsuccessful and indicates that the port is unavailable, that is, the port is closed, then a mark is made in the database of sent and received messages and the software modules on this computer systems goes into the standby state of the packet from it. If the explored computer systems is off, in this case, make a backup of ports in turn. If the scan response is negative, then the software modules goes into the state of waiting for results from such a computer systems; otherwise, a result of such a result is being tested with other active software modules that have already been generated by the distributed multi-level system.

Evaluation of the status of the software module and its verification between the rest of the software module distributed multi-level system at the stage of the exchange of messages. Processing of events for the software module, which sent packets to all computer systems: for the sent packet the response was not received; After the port is scanned, the response in the form of a result is not received or received, which is open. Then the software module monitors its result with the results of the rest of the computer systems. For this purpose, a request is made in the form of a package of the rest of the software modules, besides the subject, who must send a confirmation of their work, on the state of the program module under study. In order to carry out the research on the instructions of one selected software module, the software module is distributed, the rest of the software module sends him one packet of his condition and processes the replies from him. Submit the research results of the selected software module to the requested module. Processing the results of the study of the selected software module. If the software modules received the same response from the module under study as the software module that activated this validation event, then all the software module of the distributed multi-level system consider that the module under study is not yet active and continue to wait for the packet from it when the computer systems is turned on. The explored computer systems can be excluded, then all the software module will receive the same response. If program modules received different responses from the study module or received a certain part and the other received no other answers, as well as the software module

that activated this validation event, then all program modules notify the administrator about this event, issue a message to their computer systems screen, write in their registry of extraordinary situations and reduce the distributed multi-level system for one software module.

Determine the state of a distributed multi-level system. After a certain time set by the administrator, program modules determine the state of the distributed multi-level system with a certain periodicity in time or upon occurrence of critical events in the *computer systems*. The software module is distributed by the state-of-the-art system by its level of security. Each software module individually after the start determines its own state and in subsequent work it changes depending on the functions performed. Preparation and formation of a package with the notification of its status to each software module system and sending it from each software module to the remaining active software module. Conduct confirmation of the successful delivery of the package to each software module from the rest of the software module.

To determine the state of safety, distributed multi-level system, we use the data at the current time from its software modules: the state of each software module from the beginning of the current start, the time spent in each state of the same software module, the levels of security in each state of each software module. Calculation by the formulas (1) and (2) of the state is allowed by the distributed multi-level system by each software module on the basis of the received data from all active software module. The division into the system is carried out in two stages. At the first stage, the level of defecation will be distributed by the state-of-the-art system by the formula 1:

$$R_{b,\text{РБС},1} = \frac{\sum_{l=1}^{n}(1-\sum_{s=1}^{m} k_{s,l} * p_{s,l})}{n}, \qquad (1)$$

where $R_{b,\text{РБС},1}$- the level of security distributed multi-level system, defined in the first stage, b - security designation, l - the number of the software module, n - number of software modules, distributed multi-level system, $k_{s,l}$ - the threat factor to be affected by the software modules, s - the status of the software modules, $[0;1]$ - the value of which is determined from the segment, depending on which functional imposition is laid down in a certain state, $p_{s,l}$ - the probability of being affected, m - number of states of the software modules.

By the formula 2 distributed multi-level system determines its center at the moment, as well as, based on this value, the allocation of critical software modules.

$$\mathrm{g}\left(\mathrm{R}_{\mathrm{b,РБС,1}}, \mathrm{k}, \mathrm{s}, \mathrm{s}_{\mathrm{c,РБС}}\right) = \begin{cases} 0, & \text{if the condition is fulfilled 1} \\ 1, & \text{if the condition is fulfilled 2}, \\ 2, & \text{if the condition is fulfilled 3} \end{cases} \qquad (2)$$

where $g\left(R_{b,\text{РБС},1}, k, s, s_{\text{c,РБС}}\right)$ - the function of determining the further steps for distributed multi-level system, $R_{b,\text{РБС},1}$ - security level distributed multi-level system , received in the first stage by the formula 1, k - number of active software modules from the total number n, s - state number, $s = 1, 2, \ldots, m$, m - number of states of the software modules, $s_{\text{c,РБС}}$ - average value for distributed multi-level system based on a set of states of its software modules. The conditions for specifying the function g are presented in the table. The total number of such cases can be 64, since there are four cases for the level of security, two cases for the number of software modules, which are included in the center distributed multi-level system at the current time, eight for attributing the center to one of the states due to the study of its deviation.

Under condition 1, if $g\left(R_{b,\text{РБС},1}, k, s, s_{\text{c,РБС}}\right) = 0$, then distributed multi-level system continues to work in the mode when its software modules works in the states in which they were. In this case, no action is taken on the handling of situations in certain selected computer systems.

When the condition 2 is fulfilled, if $g\left(R_{b,\text{РБС},1}, k, s, s_{\text{c,РБС}}\right) = 1$, then distributed multi-level system continues to work in the mode when its software modules works in the states in which they were. And also, the distributed multi-level system immediately notices program modules for which additional clarification is required regarding the tasks that are performed at the current time.

When the condition 3 is fulfilled, if $g(R_{b,\text{РБС},1}, k, s, s_{\text{с,РБС}}) = 2$, then distributed multi-level system goes to the second stage of refinement of its state based on the involvement of the time characteristics of the states of all the software modules.

If the likelihood of being affected by the malicious software will affect not only the software modules or their impact on these modules is insignificant, it does not allow you to determine the state of the distributed multi-level system as critical. Such a case is possible when the study in the first stage, due to the definition of the average value and its subsequent use, was low due to the short time from the last launch of the software modules or the need for its averaging into eight states. But it may turn out that many software modules have been or were in the same state for a long time, but the use of criteria of the first phase does not distinguish them. Therefore, to take into account such boundary features, we select the probability of being affected by the malicious software for the distributed multi-level system in certain specified states and we evaluate the such cases at the second stage of the study. For the second stage of the determination of the state distributed multi-level system, the general formula 3 of the definition of the level of security will look like:

$$R_{b,\text{РБС},2} = \frac{1}{4} * \left(\sum_{s=1}^{m} \left(1 - \prod_{\substack{j=1, \\ p_{s,j}<1}}^{n} (1 - p_{s,j}) \right) * k_s + \sum_{j=1}^{n} \sum_{\substack{s=1, \\ t_{s,j}>0, \\ w_{s,j}>0}}^{m} \left(\frac{t_{s,j}}{\sum_{s=1}^{m} t_{s,j}} * \frac{w_{s,j}}{\sum_{s=1}^{m} w_{s,j}} \right) + \sum_{s=1}^{m} \left((1 + k_s) * \frac{\sum_{j=1}^{n} w_{s,j}}{\sum_{s=1}^{m} \sum_{j=1}^{n} w_{s,j}} * \frac{\sum_{j=1}^{n} t_{s,j}}{\sum_{s=1}^{m} \sum_{j=1}^{n} t_{s,j}} \right) \right), \quad (3)$$

where $R_{b,\text{РБС},2}$ - the level of security distributed multi-level system determined at the second stage, b - refer to safety, s - the number of software modules of distributed multi-level system, n - the number of software modules distributed multi-level system, m - the number of states of software modules, k_s - rate risk of being infected with malware, s that state software modules, the value of which is determined from the segment, depending on $[0; 1]$ which functional load is laid in a certain state, $p_{s,j}$ - the probability of being affected malicious software, $w_{s,j}$ - number stays with the number of software modules j in the state s, $i = 1, 2, \ldots, n$, $t_{s,j}$ - the total time spent with the number of software modules in the state. Values $p_{s,j}$ are obtained on the basis of the results of the operation of the installed in the program modules of the subsystems of the detection of certain types of malicious software.

Pinging status message distributed multi-level system of each software module with the rest of the system software modules. Analysis and processing of results each software modules of distributed multilevel system. If all software modules calculate state distributed multi-level system the same, then the system continues to work. Results of checking on the status of a distributed multi-level system match, then sent a message of each software module to all other modules. This information is stored in the internal register of events. If you find that at least one of the software modules will respond to all the rest that he received from a module result is different from his, and their, then it indicates a message for all program modules number of the software module and its result, which is different from the general results. In this situation, all the software modules, in addition to which different from other results, the team sent him to lock the computer systems and output messages cause the lock on the screen, and they block the flow of any packages from it and withdraw it from the registry system. distributed multi-level system continue to operate, but each program module will display on the screen computer systems message number and status of software modules, which removed. If you find that a software module received from all the same value of the system, but it does not match the calculated it while he sent all its value, while the software module blocks and issues outlet computer systems situations message on the screen. If parts of software modules distributed multi-level

system received packets to then start the implementation of procedures to determine the reasons for not receiving messages. After processing these events continue the work program modules.

The decision on the future work of the distributed multi-level system as a whole based on a study of the state of software modules. Determining the level of each system based on levels of software modules, their distribution by groups at risk of being affected, that is defined levels of threats and a decision on further work-based system results in function 3 and Figure 4. If the state distributed multi-level system, defined as such that the degree of safety is 0-30%, while blocking software modules to implement all of the computer systems and notify the administrator. This event may also occur upon availability in the system of a large number of software modules, which are at levels 2 or 3 for a long time. If the state distributed multi-level system, defined as having security level is 30-75%, while blocking only those make computer systems, program modules which indicates the classification of the computer systems to the level of 0-30%, to inform the administrator and transfer system status on the amount of computer systems remaining. If the state distributed multi-level system, defined as having security level is 75-100%, while they explore the computer systems, in which the security level is less than 75% for a long time. If you exceed the time limit expired to combat the threat, then make blocking software modules of the computer systems, inform the administrator and proceed without remote software modules. If the state distributed multi-level system, defined as having security level is 75-100% and after study of the computer systems, in which the security level is less than 75% for a long time, they are not found, then continue.

Removing the active software from distributed multi-level system as a result of disabling the computer systems. If one of the computer systems turns off, then its program module tells the rest of the modules and only then there is a shutdown.

Events that activate methods for detecting malware affect the state of the state of the software modules distributed multi-level system. Investigating other computer systems for the presence of similar activities and sharing the results. When switching to the software modules level 2, the method of detecting file malware based on the agent approach and the fuzzy conclusion and the method is based on the distribution of access in the network and the attraction of additional computing components of the network. When switching the software module to level 3, the method of detecting botnets or the method of detecting exploits is used.

Processing and optimization of statistical data accumulated in the system by each module separately. With a small load of computer systems and the absence of other tasks that are related to the need to stay program module at level 2 or 3, and long-term in the state of 1, the software modules goes to level 4 and researches the accumulated statistical data. In particular, it monitors and analyzes the launch procedure of the software modules in comparison with other computer systems programs and the time it starts for a certain period of time. Processing such data includes the calculation of the main statistical parameters: the definition of the average value, the variance and the mean square deviation. Implementation of optimization of accumulated statistical data in the bases of software modules. If there is a calculation and a significant deviation is detected, data optimization is not performed, the computer systems is blocked and the administrator notifies.

Knowledge sharing in the middle of a distributed multilevel system. The results obtained by one software module distributed multi-level system, which relate to the detection and localization of the malicious software, are generated in a package and sent to other software modules on the network that use these results to verify their computer systems.

Compatible completion of the match problems distributed multi-level system. Collective execution of the tasks related to the detection of the malicious software by increasing the computational resources for the software module by sending a part of the tasks to other computer systems for the investigation of the malicious software, in which there are suspicious behaviors. In particular, the involvement of other computer systemsprocessors in the software modules of the processor's work to encourage the manifestation of the malicious software. Preparation and submission to other software modules of the results obtained.

The work is distributed multi-level system, consisting of only one software module. If the distributed multi-level system remains in one of the software modules, due to the

correct completion of the work of the other software modules, then it goes to the limited use of its capabilities and can move from the first level to the other, with limited capabilities. After the next new launch, the system is in this software modules, it is compulsory to check its status, provided it goes to level 2, remaining one in the system. Thus, every last running software modules during a new run is distributed by the host system so is checked.

Replacement distributed multi-level system with new modules. Computer systems that will turn on later will rebuild the system by expanding it. Each software module of the computer systems will mail packets to other software modules.

The use of the developed method allows organizing the maintenance of the integrity of the system and the transfer of knowledge received by separate structural components of a decentralized distributed system software modules to other components. The developed method is the basis for developing a bundle of software for a decentralized distributed detection system for malicious software in local computer networks based on its security level.

The developed method of interaction of components of the system allows to support the work of distributed multi-level systems and their integrity in local area networks. Based on it, a distributed, multilevel system is developed that allows it to be filled with the implementation of various methods for detecting malicious software and apply a network component to them when detected.

Experiments. A developed method for the interaction of components distributed multi-level system and implemented methods for detecting file malicious software was tested experimentally. The experiment was conducted on a local network. The results of the system have been saved to log files. To carry out the experiment, a network of 20 computer systems was involved. Each computer system was equipped with a virtual environment based on Qemu, which was activated by the software module of the developed system for investigating allegedly malicious behavior and receiving API functions calls. The study of executable programs was carried out in three stages of their functioning: access to the computer systems, activation and implementation of the established functions. Each software module has used the database of behavioral models of the malicious software program at its various stages of operation. In order to calculate the authenticity of the malware file detection, the following experiment was conducted with different types of file malware: file (simple) viruses, polymorphic viruses, metamorphic viruses, and Trojan horses. There were generated 600 software objects with the functional load of the four types of files under consideration, each containing 150 malicious software. Each of the generated variations of metamorphic viruses used the main techniques of confusing the code: inserting garbage commands, using equivalent instructions, and moving the instruction blocks. Signatures of the generated file malicious software are absent in the databases of the signal round. All program objects were divided into groups for specifying how they would enter the computer systems to take into account all possible ways of penetration into the computer systems:

1) program objects copied to the hard disk of each computer systems;

2) program objects loaded on flash drives and connected to each computer systems;

3) program objects downloaded to a pre-created web site;

4) program objects archived and sent to previously created electronic addresses;

5) program objects are downloaded to pre-created ftp-servers of all computer systems.

The launch of the generated program objects was carried out by a special program, which was installed in each computer systems and launched one software object from the malicious software in each computer systems at a time. Then everything was repeated to select another software object. Running useful programs in all computer systems was not performed. After turning on all the computer systems, the operation systems and all the programs that are scheduled to start automatically are loaded. All computer systems contained the same hardware and software.

The results of the conducted experiment and the assessment of the authenticity of the malware detection Distributed Multilevel System [16], in which the methods are implemented, is presented in Table 1. In addition, according to the results of the experiment, the number of software module that were involved in the study throughout the experiment and the number of software module that were blocked by the rest of the software module of the developed system during the detection were

also determined. This confirms the use of the rest of the components of the distributed system in the process of detecting the malicious software of individual software module.

Table 1

Experiment results for malicious software misleading the software

Software objects with explicit inside malicious software		Number of programs detected as suspicious	Percent detection,%	The number of software modules that were involved in the study of the whole experiment	Number of software modules that were blocked by the rest of the software modules of the developed system during the detection
File viruses	150	146	97,3%	0	2
Polymorphic viruses	150	134	89,3%	57	7
Metamorphic viruses	150	138	92,3%	24	3
Trojan programs	150	128	85,3%	2	5
Total	600	546	90,9	20,75	5,7

The following known anti-virus tools were selected for benchmarking: ESET Smart Security (version 10.1.204.0), Avast (version 17.5.2303), Comodo Antivirus (version 8.2.0.4674), Kaspersky (version 17.0.0.61), McAfee Internet Security (version 10.1.0), Dr.Web (version 11.0), Microsoft Security Essentials (version 4.11.15063.446), Avira Antivirus (version 10.0). The results of the developed Distributed Multilevel System for detecting malicious software are presented in the diagram in Fig. 2.

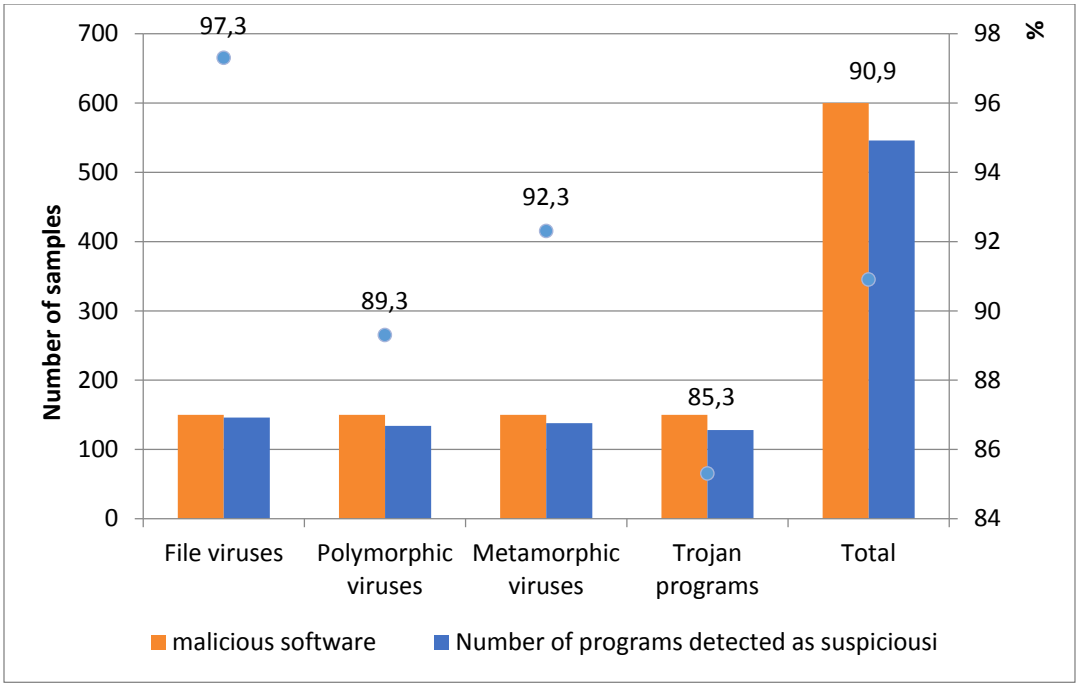

Fig. 2. Experiment results

The results of experimental studies using the developed Distributed Multilevel System based on the method of organizing the interaction of its components are confirmed by the fidelity of the scientific principles of the developed methods and the efficiency of the architecture of the distributed multilevel system. Its implementation increases the authenticity of the detection by 5-12% in the network representation compared to the host, and by 2-4% compared with the existing network firewall detecting the file new malware.

Conclusions. Using the developed method of interaction of components of the system allows organizing the support for the integrity of the distributed multi-level system and the transfer of knowledge acquired by the individual structural components of the system software modules of the remaining components. The developed method is the basis for developing a bundle of software distributed multi-level decentralized system of detection of malicious software.

The direction of further research is the development of new models of malicious software, the detailed structure of the distributed multi-level system, its states and filling subsystems of detection of various types of malware.

REFERENCES:

1. Security Response Publications. (2019). Monthly Threat Report. Retrieved from https://www.symantec.com/security_response/publications/monthlythreatreport.jsp.
2. McAfee Labs. (2019). McAfee Labs Threat Report. December 2017. Retrieved from https://www.mcafee.com /us/resources/reports/rp-quarterly-threats-dec-2017.pdf.
3. Symantec. (2019). Overview of Symantec Endpoint Protection 12. Part 2. Retrieved from https://www.anti-malware.ru/reviews/Symantec_Endpoint_Protection_12_2.
4. Palo Alto Networks. (2019). Retrieved from https://www.paloaltonetworks.com/
5. Malwarebytes. (2019). Malwarebytes Endpoint Security. Retrieved from https://ru.malware bytes.com/business/endpointsecurity/
6. Cisco. (2019). Cisco NAC Appliance (Clean Access). Retrieved from https://www.cisco. com/c/en/us/products/security/nac-appliance-clean-access/index.html.
7. Comodo (2019). Comodo CyberSecurity. Retrieved from https://www.comodo.com/
8. Branitskiy, A., Kotenko, I. (2017). Hybridization of computational intelligence methods for attack detection in computer networks. *Journal of Computational Science*, 23, 145–156.
9. Bezobrazov, S., Sachenko, A., Komar, M., Rubanau, V. (2016). The methods of artificial intelligence for malicious applications detection in Android OS. *International Journal of Computing*, 15 (3), 184–190.
10. David, B., Filiol, E., Gallienne, K. (2017). Structural analysis of binary executable headers for malware detection optimization. *Journal of Computer Virology and Hacking Techniques*, 13 (2), 87–93.
11. Eslahi, M., Abidin, W. Z., Naseri, M. V. (2017). Correlation-based HTTP Botnet detection using network communication histogram analysis. In *Proceedings of 2017 IEEE Conference on Application, Information and Network Security*, Miri, Malaysia, 2017 (pp. 7–12).
12. Pronoza, A., Vitkova, L., Chechulin, A., Kotenko, I. (2019). Visual Analysis of Information Dissemination Channels in Social Network for Protection Against Inappropriate Content. In *Proceedings of the Third International Scientific Conference: Intelligent Information Technologies for Industry*, *Volume 2*, Sochi, Russia, 2019 (pp.95–105).
13. Sun, M., Xu, G., Zhang, J., Kim, D. (2017). Tracking you through DNS traffic: Linking user sessions by clustering with Dirichlet mixture model. In *Proceedings of 20th ACM International Conference on Modeling, Analysis, and Simulation of Wireless and Mobile Systems*, Miami, FL, US, 2017 (pp. 303–310).
14. Schomp, K., Rabinovich, M., Allman, M. (2016). Towards a model of DNS client behavior. In *Proceedings of the International Conference on Passive and Active Network Measurement, volume 9631*, Heraklion, Crete, Greece, 2016 (pp. 263–275).
15. Zheng, J., Li, Q., Gu, G., Cao, J., Yau, D. KY, Wu, J. (2018). RealtimeDDoS Defense Using COTS SDN Switches via Adaptive Correlation Analysis. *IEEE Transactions on Information Forensics and Security*, 13(7), 1838–1853.
16. Markowsky, G., Savenko, O., Sachenko, A. (2019). Distributed Malware Detection System Based on Decentralized Architecture in Local Area Networks. *Advances in Intelligent Systems and Computing III*, 871, 582–598.

САВЕНКО Олег Станіславович
к.т.н., професор, декан факультету програмування та комп'ютерних і телекомунікаційних систем програмування,
E-mail: savenko_oleg_st@ukr.net

ОРГАНІЗАЦІЯ ВЗАЄМОДІЇ КОМПОНЕНТІВ РОЗПОДІЛЕНОЇ БАГАТОРІВНЕВОЇ СИСТЕМИ ВИЯВЛЕННЯ ЗЛОВМИСНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ НА ОСНОВІ РІВНІВ ЇХ БЕЗПЕКИ

У роботі представлено розроблений метод взаємодії компонентів розподіленої багаторівневої системи виявлення зловмисного програмного забезпечення (ЗПЗ) на основі децентралізованої та самоорганізованої архітектури в локальних мережах. Її особливістю є синтез в ній вимог розподіленості, децентралізованості, багаторівневості та самоорганізованості. Це дозволяє використовувати її автономно. Основою побудованої розподіленої системи є її структурні компоненти, які представляються автономними програмними модулями, що можуть перебувати в різних станах. Перехід між станами модулів здійснюється на основі визначеної множини переходів. Взаємодія та спілкування між автономними програмними модулями базується на основі їх перебування в певних станах під час експлуатації та встановлюється правилами розробленого методу. Розподілена система є реагуючою системою, яка здійснюватиме моніторинг визначених подій. Кожен програмний модуль містить резидентний механізм, рушійні механізми для переходу між станами, переходи між якими задаються підмножинами переходів, данні для яких формуватимуться з використанням технологій штучного інтелекту. Крім того, особливістю компонентів системи є така самоорганізація, що дає змогу здійснювати обмін знаннями в середині системи, яка на відміну від відомих систем дозволяє використовувати знання отримані окремими частинами системи в інших частинах. Розроблена система дозволяє здійснювати її наповнення підсистемами виявлення різного типу зловмисного програмного забезпечення в локальних обчислювальних мережах. Метод взаємодії компонентів розподіленої багаторівневої системи виявлення ЗПЗ встановлює порядок здійснення комунікації між частинами системи та обміну знаннями між ними. Він застосовуватиметься для організації взаємодії компонент системи і підтримки її цілісності. Для вирішення проблеми з безпосереднього виявлення ЗПЗ в локальних обчислювальних мережах застосовуватимуться методи, які відноситимуться до нижчого рівня системи, що включатимуть архітектурні особливості розподіленої системи і технології виявлення ЗПЗ. Проте розроблений метод взаємодії включає можливість визначення стану розподіленої багаторівневої системи в залежності від станів окремих модулів та на його основі згідно нього прийматимуться рішення про подальшу роботу системи в цілому і її конфігурацію. Метод регламентує дії тієї частини системи, яка відноситься до зв'язуючого програмного забезпечення розподіленої системи. Проведені експерименти з використання розробленої розподіленої системи показали можливість залучення до виявлення ЗПЗ обчислювальних потужностей інших КС локальної мережі. Отримані результати експериментів показують підвищення достовірності виявлення ЗПЗ.

Ключові слова: *зловмисне програмне забезпечення, розподілена багаторівнева система, децентралізована система, комп'ютерні системи, локальна мережа*

САВЕНКО Олег Станиславович
к.т.н., профессор, декан факультета программирования, компьютерных и телекомуникационных систем, E-mail: savenko_oleg_st@ukr.net

ОРГАНИЗАЦИЯ ВЗАИМОДЕЙСТВИЯ КОМПОНЕНТОВ РАСПРЕДЕЛЕННОЙ МНОГОУРОВНЕВОЙ СИСТЕМЫ ОБНАРУЖЕНИЯ ВРЕДОНОСНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ НА ОСНОВЕ УРОВНЕЙ ИХ БЕЗОПАСНОТИ

В работе представлен разработанный метод взаимодействия компонентов распределенной многоуровневой системы обнаружения вредоносного программного обеспечения (ВПО) на основе децентрализованной и самоорганизующейся архитектуры в локальных сетях. Ее особенностью является синтез в ней требований распределенности, децентрализованости, многоуровневости и самоорганизованности. Это позволяет использовать ее автономно. Основой построенной распределенной системы является ее структурные компоненты, которые представляются автономными программными модулями, которые могут находиться в разных состояниях. Переход между состояниями модулей осуществляется на основе определенной множества переходов. Взаимодействие и общение между автономными программными модулями базируется на основе их пребывания в определенных состояниях при эксплуатации и устанавливается правилами разработанного метода. Распределенная система является реагирующей системой, которая будет осуществлять мониторинг определенных событий. Каждый программный модуль содержит резидентный механизм, движущие механизмы для перехода между состояниями, переходы между которыми задаются подмножествами переходов, данные для которых будут формироваться с использованием технологий искусственного интеллекта. Кроме того, особенностью компонентов системы есть такая самоорганизация, что позволяет осуществлять обмен знаниями внутри системы, которая в отличие от известных систем позволяет использовать знания, полученные отдельными частями системы в других частях. Разработанная система позволяет осуществлять ее наполнения подсистемами обнаружения различного типа вредоносных программ в локальных вычислительных сетях. Метод взаимодействия компонентов распределенной многоуровневой системы обнаружения ВПО устанавливает порядок осуществления коммуникации между частями системы и обмена знаниями между ними. Он будет применяться для организации взаимодействия компонент системы и поддержания ее целостности. Для решения проблемы с непосредственного обнаружения ВПО в локальных вычислительных сетях применяться методы, которые будут относиться к более низкому уровню системы, включающие архитектурные особенности распределенной системы и технологии обнаружения ВПО. Однако разработанный метод взаимодействия включает возможность определения состояния распределенной многоуровневой системы в зависимости от состояний отдельных модулей и на его основе по нему будут приниматься решения о дальнейшей работе системы в целом и ее конфигурации. Метод регламентирует действия той части системы, которая относится к связующему программному обеспечению распределенной системы. Проведенные эксперименты по использованию разработанной распределенной системы показали возможность привлечения к выявлению ВПО вычислительных мощностей других КС локальной сети. Полученные результаты экспериментов показывают повышение достоверности обнаружения ВПО.

Ключевые слова: *вредоносное программное обеспечение, распределенная многоуровневая система, децентрализированная система, компьютерные системы, локальная сеть*

ЛІТЕРАТУРА:

1. Security Response Publications. Monthly Threat Report. URL: https://www.symantec.com/security_response/publications/monthlythreatreport.jsp. (Last accessed: 07.04.2019).
2. McAfee Labs. McAfee Labs Threat Report. December 2017. URL: https://www.mcafee.com /us/resources/reports/rp-quarterly-threats-dec-2017.pdf (Last accessed: 07.04.2019).

3. Overview of Symantec Endpoint Protection 12. Part 2. URL: https://www.anti-malware.ru/reviews/Symantec_Endpoint_Protection_12_2 (Last accessed: 07.04.2019).
4. Palo Alto Networks. URL: https://www.paloaltonetworks.com/ (Last accessed: 07.04.2019).
5. Malwarebytes Endpoint Security. URL: https://ru.malwarebytes.com/business/ endpointsecurity/ (Last accessed: 07.04.2019).
6. Cisco NAC Appliance (Clean Access). URL: https://www.cisco.com/c/en/us/products/ security/nac-appliance-clean-access/index.html (Last accessed: 07.04.2019).
7. Comodo CyberSecurity. URL: https://www.comodo.com/ (Last accessed: 07.04.2019).
8. Branitskiy A., Kotenko I. Hybridization of computational intelligence methods for attack detection in computer networks. *Journal of Computational Science*, 2017. No. 23. P. 145–156.
9. Bezobrazov S., Sachenko A., Komar M., Rubanau V. The methods of artificial intelligence for malicious applications detection in Android OS. *International Journal of Computing*, 2016. Vol. 15, No. 3. P. 184-190.
10. David B., Filiol E., Gallienne K. Structural analysis of binary executable headers for malware detection optimization. *Journal of Computer Virology and Hacking Techniques*, 2017. Vol. 13, No. 2. P. 87–93.
11. Eslahi M., Abidin W. Z., Naseri M. V. Correlation-based HTTP Botnet detection using network communication histogram analysis. *The 2017 IEEE Conference on Application, Information and Network Security:* Proceedings (Miri, Malaysia, November 13-14 2017). Miri, 2017. P. 7–12.
12. Pronoza A., Vitkova L., Chechulin A., Kotenko I. Visual Analysis of Information Dissemination Channels in Social Network for Protection Against Inappropriate Content. *The Third International Scientific Conference: Intelligent Information Technologies for Industry*, *Volume 2*: Proceedings (Sochi, Russia, September 17-21 2019). Sochi, 2019. P. 95–105.
13. Sun M., Xu G., Zhang J., Kim D. Tracking you through DNS traffic: Linking user sessions by clustering with Dirichlet mixture model. *The 20th ACM International Conference on Modeling, Analysis, and Simulation of Wireless and Mobile Systems*: Proceedings (Miami, FL, US, November 24–26 2017). Miami, 2017. P. 303–310.
14. Schomp K., Rabinovich M., Allman M. Towards a model of DNS client behavior. *International Conference on Passive and Active Network Measurement, volume 9631*: Proceedings (Heraklion, Crete, Greece, 31 March – 1 April 2016). Heraklion, 2016. P. 263–275.
15. Zheng J., Li Q., Gu G., Cao J., Yau D. KY, Wu J. RealtimeDDoS Defense Using COTS SDN Switches via Adaptive Correlation Analysis. *IEEE Transactions on Information Forensics and Security*, 2018. Vol. 13, Issue 7. P. 1838–1853.
16. Markowsky G., Savenko O., Sachenko A. Distributed Malware Detection System Based on Decentralized Architecture in Local Area Networks. *Advances in Intelligent Systems and Computing III*, 2019. Vol. 871. P. 582–598.