ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

UDC 004.491.052.42 **doi: 10.32620/reks.2023.4.10**

Antonina KASHTALIAN[1], Sergii LYSENKO[1], Bohdan SAVENKO[1], Tomas SOCHOR[2], Tetiana KYSIL[1]

[1] ***Khmelnitsky National University, Khmelnitsky, Ukraine***
[2] ***Prigo University, Havirov, Czech Republic***

PRINCIPLE AND METHOD OF DECEPTION SYSTEMS SYNTHESIZING FOR MALWARE AND COMPUTER ATTACKS DETECTION

The number of different types and the actual number of malware and computer attacks is constantly increasing. Therefore, detecting and counteracting malware and computer attacks remains a pressing issue. Users of corporate networks suffer the greatest damage. Many effective tools of various kinds have been developed to detect and counteract these effects. However, the dynamism in the development of new malware and the diversity of computer attacks encourage detection and countermeasure developers to constantly improve their tools and create new ones. The object of research in this paper is deception systems. The task of this study is to develop the elements of the theory and practice of creating such systems. Deception systems occupy a special place among the means of detecting and counteracting malware and computer attacks. These systems confuse attackers, but they also require constant changes and updates, as the peculiarities of their functioning become known over time. Therefore, the problem of creating deception systems whose functioning would remain incomprehensible to attackers is relevant. To solve this problem, we propose a new principle for the synthesis of such systems. Because the formation of such systems will be based on computer stations of a corporate network, the system is positioned as a multi-computer system. The system proposes the use of combined baits and traps to create false attack targets. All components of such a system form a shadow computer network. This study develops a principle for synthesizing multi-computer systems with combined baits and traps and a decision-making controller for detecting and countering IEDs and spacecraft. The principle is based on the presence of a controller for decisions made in the system and the use of specialized functionality for detection and counteraction. According to the developed principle of synthesizing such systems, this paper identifies a subset of systems with deception technologies that must have a controller and specialized functionality. The decision-making controller in the system is separate from the decision-making center. Its task is to choose the options for the next steps of the system, which are formed in the center of the system, depending on the recurrence of events. Moreover, prolonged recurrence of external events requires the system center to form a sequence of next steps. If they are repeated, the attacker has the opportunity to study the functioning of the system. The controller in the system chooses different answers from different possible answers for the same repeated suspicious events. Thus, an attacker, when investigating a corporate network, receives different answers to the same queries. Specialized functionality, in accordance with the principle of synthesis of such systems, is implemented in the system architecture. It affects the change of system architecture in the process of its functioning as a result of internal and external influences. This paper also considers a possible variant of the architecture of such deception systems, in particular, the architecture of a system with partial centralization. To synthesize such systems, a new method for synthesizing partially centralized systems for detecting malware in computer environments has been developed based on analytical expressions that determine the security state of such systems and their components. In addition, the experiments showed that the loss of 10-20% of the components does not affect the performance of the task. The results of the experiments were processed using ROC analysis and the algorithm for constructing the ROC curve. The results of the experiments made it possible to determine the degree of degradation of the systems constructed in this manner. Conclusions. This paper presents a new principle for the synthesis of multi-computer systems with combined decoys and traps and a decision-making controller for detecting and counteracting IEDs and spacecraft, as well as methods for synthesizing partially centralized systems for detecting malware in computer networks.

Keywords: *deception systems; deception systems synthesizing; principle of systems synthesis, controller, distributed systems; honeynet; trap; baits; malware detection; partial centralization.*

1. Introduction

1.1. Motivation

Malware continues to be actively developed and distributed. An important element of counteracting it is properly synthesized systems that can detect malware and counteract it by creating false attack objects.

An area of development of such systems is the development of systems whose behavioral logic and architecture are difficult or impossible for attackers to understand.

Having created such systems, they can be filled with functionality as needed, which can further position them as deception systems, network baits, and highly specialized systems for detecting a specific class of malware. Such systems include deception systems.

Due to the specifics of the tasks they are supposed to perform, the actual task is to develop new principles of their synthesis, which will allow the attacker to create new features in such systems that will be difficult for him to understand. One class of such systems is partially centralized distributed systems for detecting malware, as described in [1].

The functioning of partially centralized distributed systems in accordance with the principles of self-organization and adaptability is ensured not only by the organization of communication between their components or the implementation of certain specially oriented tasks for which they are created, but primarily by internal mechanisms, methods, and algorithms that enable such systems to solve tasks without user intervention, independently make decisions on the next steps of the system, and adapt to changes in the external environment.

1.2. Previous works

There are many various studies devoted to the problem of malware detection. Despite the large number of different methods for detecting and mitigating cyberattacks caused by malware, the steady increase in their number confirms that this problem is not solved today.

A variant of the architecture of partially centralized systems for detecting malicious software in computer networks is presented in [1]. The feature of the described architecture is that it should enable such systems to function according to the principles of self-organization and adaptability. This will give them opportunities to determine their next steps in the process of unionization.

Such systems can be used to counter and detect malware.

The systems synthesized in this way function as centralized, but their decisions are made in part of the components defined by the system in a decentralized manner. To implement the internal mechanism of their functioning, it is necessary to develop a method for organizing their functioning.

The peculiarity of the synthesized system, which is related to its centralization, decentralization, and hybrid architecture, concerns the center of the system. This class of systems can be specified according to the principle of their synthesis, which requires appropriate development.

1.3. State of the art

To use deception technology, various types of baits and traps [2] have been developed that mimic the operation of real systems. The market offers several solutions based on the use of deception technology and malware. Let's take a look at the characteristic features of such systems.

The main features of the Acalvio ShadowPlex system [3] are the patented architecture of the deception farm and autonomous deception. This system automates and simplifies the configuration and deployment of baits and traps using predefined deceptive objects and objects that are generated and placed by the system based on recommendations and artificial intelligence. It supports a significant number of deceptive objects, including baits that mimic hosts running operating systems, including IoT hosts, endpoint baits, fake registry entries, credentials, and shared disks.

One of the first systems to use deception technology to add response capabilities was the Attivo ThreatDefend Deception and Response Platform [4]. The system can be deployed locally, in the cloud, in data centers, or in a hybrid environment. Similar to other systems, deceptive objects are designed to identify intruders trying to access the network and data. This system not only detects access attempts but also ensures that the deceptive object interacts with the attacker, simulating the reaction that the attacker can expect from real objects. In other words, simultaneously with network protection, the study of malicious intentions and tactics is ensured.

The Proofpoint Identity Threat Defense system [5] creates a deceptive environment for the attacker. The agentless architecture prevents attackers from detecting deceptive objects. The system detects any changes in the environment and activates deceptive capabilities to ensure that the attacker is stopped before gaining access to corporate network resources. Protection against attacks and early response is provided for email services, mobile communications, social networks, and desktop workstations.

The CounterCraft Cyber Deception Platform [6] system uses active baits to detect intruders. These baits can be deployed as endpoints, servers, or in online platforms and are flexible to customize. The system is designed to facilitate online interaction with attackers. The system provides the ability to deploy quickly and control based on data collected in the environment by agents.

The Fidelis Deception platform [7] allows the user to quickly and dynamically create a deceptive environment that contains baits and traps for user applications, services, network connections, integrated credentials of the active directory, memory, endpoints, and servers.

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

All actions that occur in the deceptive environment are tracked and available to the administrator to make decisions about studying the actions of intruders, neutralizing attacks, and protecting against them. The advantages of the Fidelis Deception platform are as follows: automatic creation of a realistic deception environment that includes baits and traps of various types; building real operating system baits along with simulated baits covering the entire corporate network; fast deployment and efficiency from the start; high accuracy with no false positives; constant updating of the intelligent deception environment; and proactive protection.

The CommVault system [8] provides several data management solutions for data protection, management, and optimization. In the area of cybersecurity, CommVault offers risk analysis and full data protection, including auto-recovery, threat scanning, active directory, and database protection. The system's modules combine early warning capabilities with rapid response capabilities to neutralize attacks in time before damage can occur. "Zero-day" attacks are also detected and neutralized that are difficult or impossible to detect using traditional technologies. The system also makes it possible to immediately detect and neutralize hidden attacks that are spreading in the network environment. In other words, this system enables identification and prevention of attacks even before they start.

The Labyrinth Deception Platform [9, 10] creates a deceptive environment that simulates the services and content of the real part of the network. The solution is based on the so-called Points, which are intelligent hosts that simulate software, content, routers, and devices. These deception points detect malicious activity within the corporate network, providing comprehensive protection against possible attacks. The deceptive environment encourages attackers to take actions that allow detection and tracking of their activity.

Bait, traps, and generally false targets for attacks in corporate networks can vary. Let's consider some of the research papers that discuss some types of such false attack targets and methods of their organization and use.

The article [11] proposes a new type of decoy system based on deception security technology. The dynamic deception method is adapted to collect unused IP addresses in the network. The security properties of deceptive elements are tested in the study [12] with the help of attackers using reinforcement learning. For testing, deceptive elements are included in the Microsoft CyberBattleSim research environment. The success of attackers depends on the number and location of deceptive elements. The purpose of cyber deception is to distort the state of the network to mislead attackers, falsify their conclusions, and distract them from their goals. The article [13] proposes a two-phase deception method based on bait localization. In the first phase, a proactive decoy localization policy is developed, and in the second phase, a reactive deception approach is proposed that dynamically determines the location of baits according to updates of the intrusion detection system. Thus, the defense system partially tracks the activity of the attacker.

The strategy [14] for locating baits in the network should consider not only aspects of the protected network but also the preferences of attackers. To achieve this goal, we propose a game-theoretic method that generates an optimal decoy placement strategy in accordance with an attack-defense scenario. The study [15] proposes a new method for cyber-manipulation using decoy localization and software diversity to improve network security. The study [16] proposes a scalable algorithm for placing baits over an attack graph. The authors express a two-person zero-sum strategic game between a defender and an attacker. This formulation reflects the importance of different nodes within the network.

When using baits, certain compromises must be made [17]. On the one hand, decoy systems and services must be relevant and attractive to the attacker, and on the other hand, computational and related costs must be consistent with the functional and budgetary constraints of the system. Therefore, it is impossible to create a single, unchanging decoy configuration for different types of systems and to consider all possible types of attackers.

Detecting malicious packets among a significant amount of normal activity is time-consuming [18]. The range of vulnerabilities is expanding with the development of technologies such as IoT, industrial automation, CPS, and digital twins. Baits are used in malicious packet identification to eliminate false positives. In addition to analyzing and reporting intrusion patterns, they are also used to prevent access to operational devices by mimicking real systems operating on the network and capturing and detaining attackers. Baits in computer networks are effective when they deceive cybercriminals in such a way that they do not consider themselves to be real decoys [19]. Therefore, to make decoy deception more effective, it is necessary to apply it in a more diverse way. Much of the critical data of organizations is now stored in databases, which is an attractive target for attackers [20]. The introduction of digital technologies and the increase in the number of connections between organizations, together with the growing complexity of information systems, leads to an increase in the spread of attacks. At the same time, the improvement of attackers'' skills leads to more "sophisticated" attacks.

For researchers in the field of cybersecurity [21], the question remains how to eliminate virtual machine artifacts to effectively build deceptive "baits" for col-

lecting and analyzing malware. The method of using Linux containers for this purpose is investigated. Today, the reactions of computer systems are, in most cases, predictable [22], which provides attackers with information on how to access them. It has been shown that deception technologies are used in many successful computer hacks, including phishing, social engineering, and drive-by-downloads attacks.

Paper [23] provides an overview of the problem of baits and defense strategies based on deception technology. The author defines the phenomenon of baits and summarizes their advantages and disadvantages, and their legal and ethical aspects. Baits are classified into different categories, and examples of baits that are actively being developed and those that have had a significant impact are presented. Baits are designed to distract attackers from computer resources [24]. Baits also track attacker activity and help researchers study attack patterns. However, baits can also be identified by attackers using various identification methods. A decoy [25] is a tool with an isolated and separated network that mimics a real network of value to attackers. It can be seen as a fake system that looks like the real thing and aims to attract attackers, interact with them, and monitor the interaction between the attackers and the infected device. At the same time, baits are becoming an important entity for information cybersecurity researchers to recognize attacks and in deception technology. Today, a significant number of devices are connected to the Internet [26], which increases the need to protect them from cyberattacks. The decoy-based deception mechanism is considered to be a method to ensure the security of modern IoT networks.

The tactic of confusing the attacker is presented in [27]. A self-adaptive system that incorporates resilience mechanisms is presented in [28]. The analysis of computer attacks and malware in terms of the implementation of the developed methods in detection systems is presented in [29, 30]. An important feature in the development of detection methods is to take into account the peculiarities of their implementation directly in systems [31].

In [32], a model and training method for malware traffic detection based on a decision tree ensemble is presented. methods and technologies for ensuring cybersecurity of industrial and web-oriented systems and networks are presented in [33]. Another cyber attack detection system based on information-extreme machine learning is presented in [34]. In the study [35] an overview of cyber threats and vulnerabilities is presented. The article [36] presents the consistency issue and related trade-offs in distributed replicated systems and databases. The study [37] presents a method for classifying malware using images that use dual attention and convolutional neural networks. In [38], state-of-the-art malware classification approaches are presented. The study [39] proposes an unsupervised deep learning approach that employs an artificial neural network to detect anomalies in an insider cyber security attack scenario.

Thus, the use of false decoy and trap attacks is a promising and actively developing area of research. The creation and management of such false objects requires the development of a distributed system for operation in a corporate network, which would organize the functioning of the entire system at the levels of interaction of its components, decision-making, and autonomous operation of individual components. In this regard, the principle of synthesizing such systems using false attack objects and, accordingly, sets of baits and traps needs to be developed.

1.4. The purpose and tasks of research

From the above review of literature sources, it follows that the following task needs to be solved: the development of a new principle for synthesizing deception systems and a method for synthesizing partially centralized distributed systems.

The aim of this paper is to develop a principle of multi-computer deception systems for malware and computer attack detection based on baits and traps and to develop a method of creating partially-centralized systems as a class of deception system. Such systems should confuse attackers, which will improve the effectiveness of countering malware and cyberattacks.

The paper structure is as follows.

Section 1 is devoted to previous work. Section 2 presents the Related works section – a brief analysis of the very modern and the latest ideas and methods addressed to solve the problem of IoT malware detection with its advantages and disadvantages. Sections 3 and 4 discuss the main idea of the research: the development of the principle of synthesis of multicomputer systems of combined bait and traps and the method of creating partially centralized systems for detecting malware in computer networks. Section 5 describes the experimental results of this research. In addition, conclusions present the obtained results of the research.

3. Principle of synthesis of multi-computer systems using combined anti-virus bait and traps and the decision-making controller for detecting malware and computer attacks in corporate networks

Users of computer networks need systems for detecting malicious software and computer attacks that will allow, in addition to ensuring security at various

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

stages of possible penetration of computer systems or stations that are connected to the network, for the stage when at all previous stages such detections were not made, but penetration of the system could have occurred. Among the systems for detecting malware and computer attacks, there are systems that, in addition to detecting threats, create false targets for attacks in computer networks, which allows administrators of such networks to monitor processes in networks that are malicious or abnormal and need to be stopped. Therefore, systems focused on the detection of malware and computer attacks that have passed certain stages of protection, which used traditional means and systems of prevention, detection, and counteraction, the purpose of which and possible configuration options for use are known to attackers, are promising for development. Among such systems, a special place in the classification is occupied by prevention, detection, and counteraction systems with a certain set of baits and traps for malware and computer attacks. Their use creates false attack targets for the attacker and allows the information about such attacks and the spread of malware in computer stations in the network to be saved.

To improve the effectiveness of systems for detecting and counteracting malware and computer attacks through the use of baits and traps, it is necessary to integrate these tools into complex systems involving all computer stations in the network and organize their operation in such a way that they can jointly and without user intervention respond to malicious and anomalous processes. Thus, it is necessary to build not just one bait and trap in a particular computer station but a network of bait and traps to provide comprehensive protection of a computer network at the stage when computer attacks have managed to pass through firewalling and malware has managed to overcome scanning by antivirus tools and systems. Such a system of baits and traps can be a combined system, and to achieve an effective result, it should include shadow baits and traps that will allow you to establish and track the attacker's behavior during the attack, as well as detect malware and computer attacks with a higher probability. The effectiveness of such tools depends on the organizational component of the system.

Such a principle should define the general requirements for the construction of elements of the theory of creating multi-computer systems with combined baits and traps and a decision-making controller to detect and counteract malware and computer attacks. When describing these systems, the guiding principle encompasses the fundamental traits that enable the system to operate effectively. Without formalization and a clear demonstration of its proper functionality, the system will fail to achieve its intended purpose. In addition, the principle of synthesis of such systems will allow the formation of a class of such systems and will develop elements of the theory of multicomputer systems in terms of systems that combine specialized functionality with a decision-making controller to detect malware using combined baits and traps.

In the context of the development of elements of the theory of multicomputer systems, this principle is a systematic principle because it refers to the definition of the mechanisms of system functioning. In this case, it is necessary to specify such features and characteristic properties of the systems that reflect the smallest number of factors that will determine how the system will function.

The principle of the synthesis of multicomputer systems with combined baits and traps and a decision-making controller for detecting and counteracting malware and computer attacks is set by considering the details of the system's decision-making controller and specialized functionality for detecting malware using combined baits and traps.

The architecture of multicomputer systems, taking into account the principle of synthesis of such systems, can be centralized, decentralized, or hybrid with different degrees of centralization. Accordingly, the decision-making center of such multicomputer systems may be located in one or more components of the system, and this, like the architecture, will not affect the principle of system synthesis and its non-fulfilment. The centre can move between components depending on the current state of the system. In addition, the architecture of such systems can be flexibly rebuilt, if necessary, when the external environment changes and the system is affected, which characterizes the specifics of the tasks it performs. However, such features do not affect the requirement of the principle of synthesis of such systems. The peculiarity of the proposed principle of system synthesis is that it ensures control over the decisions made in the decision-making center, i.e., the mandatory presence of a decision-making controller. At the same time, the decision controller should be able to influence their implementation by approving or rejecting the proposed next steps of the system, as well as approving another close or alternative solution. Such features of a decision controller are required because the system is designed to perform specific tasks associated with the interaction of system components or elements with malware and computer attacks. Accordingly, attackers can repeat their actions many times in the same way, which will bring the system and its respective components to the same state. Because of such testing of the system, the attacker will be able to study its behavior and in a certain time will be able to bypass it. Therefore, the system's decision controller should influence the final decision-making by selecting the next steps of the system as a reaction to changes in the external environment and the

state of the system and its components. Such a change in the choice of the next steps of the system will lead to complications for the attacker in terms of studying the behavior of detection tools for countering malware and computer attacks in corporate networks.

The requirement to combine in such systems specialized functionality for processing events in a corporate network, i.e., spatial distribution, and the presence of a decision-making subsystem in which decisions will be developed, and their implementation is possible only after approval by the controller, establishes factors for developing the principle of synthesis of multi-computer systems with combined baits and traps and a decision-making controller for detecting and counteracting malware and computer attacks. In particular, we formalize the systems, their components, and their properties that are necessary to fulfill the requirements of the principle of system synthesis.

Let us denote by the symbol $\mathfrak{P}$ the principle of the synthesis of multicomputer systems with combined baits and traps and a decision-making controller to detect and counteract malware and computer attacks. Then, as a mapping from the entire set of multicomputer systems $\mathfrak{C}$, it will form a subset of systems $\mathfrak{S}$ for which the requirement of the principle $\mathfrak{P}$ will be fulfilled. That is, the given mapping by the formula $\mathfrak{C} \xrightarrow{\mathfrak{P}} \mathfrak{S}$ will form a class of systems with the requirements set by the principle $\mathfrak{P}$, and it is necessary to detail the components of such systems for their further synthesis. Let us define each of the defining components and properties that need to be implemented in the architecture of such systems as a subset $\mathfrak{V}_i$ (i = 1,2, … , $n_{\mathfrak{V}}$, $n_{\mathfrak{V}}$ is the number) The presence of possible variants among $\mathfrak{V}_i$ (i = 1,2, … , $n_{\mathfrak{V}}$, $n_{\mathfrak{V}}$ is the number) is acceptable. For example, such systems may be centralised, decentralised or hybrid with a certain degree of centralization, which may also provide opportunities for their division into separate types, and at the same time they will meet the requirements of the $\mathfrak{P}$ principle.

Let us consider possible variants of components and defining properties for the class of systems $\mathfrak{S}$: $\mathfrak{V}_1$is type of system architecture (centralised, decentralised, hybrid); $\mathfrak{V}_2$ is types and number of centres in the system architecture (integral in one component, divided into equivalent parts in different components, hierarchically divided in different components, integral hierarchical in different components); $\mathfrak{V}_3$ – adaptability of the system when external conditions change (change of its functioning algorithms, change of system architecture, change of functioning algorithms and change of system architecture); $\mathfrak{V}_4$ – the nature of changes in the centre of the system (change in parameter values, change in the architecture of the centre, change in parameter values and change in the architecture of the centre); $\mathfrak{V}_5$ – self-organization of the system (creation of the organization of functioning of a complex system, reproduction of the organization of functioning of a complex system, improvement of the organization of functioning of a complex system), $\mathfrak{V}_6$ – flexibility of the system (quick reconfiguration of the system under the influence of external events, latent reconfiguration of the system); $\mathfrak{V}_7$ – independence in decision-making (decision-making by the entire system centre, decision-making by a part of the system centre); $\mathfrak{V}_8$ – influence on the system (internal events, external events, internal and external events); $\mathfrak{V}_9$ – multi-agency in the system for decision-making (multi-agency, single-agency, no agents); $\mathfrak{V}_{10}$ – control of decisions in the system (presence of a controller, absence of a controller); $\mathfrak{V}_{11}$ – availability of specialised functionality in the system (formation of internal events in the system by specialised functionality based on the results of execution of the Each of the characteristics $\mathfrak{V}_i$ (i = $n_{\mathfrak{V}}$, $n_{\mathfrak{V}}$ is the number of characteristics) is a set that contains typical elements related to systems $\mathfrak{C}$. When applying the principle $\mathfrak{P}$, systems of type $\mathfrak{C}$ are synthesised. To synthesize systems according to the principle $\mathfrak{P}$, that is, the formation of the set $\mathfrak{C}$ according to the definition of the direct product of sets $\mathfrak{V}_i$ (i = $n_{\mathfrak{V}}$, $n_{\mathfrak{V}}$ is the number of characteristics) is as follows:

$$\mathfrak{S} = \{(\mathfrak{v}_1, \mathfrak{v}_2, \dots, \mathfrak{v}_{11}) | (\mathfrak{v}_1, \mathfrak{v}_2, \dots, \mathfrak{v}_{11}) \in \in \mathfrak{V}_1 \times \mathfrak{V}_2 \times \dots \times \mathfrak{V}_{10,1} \times \mathfrak{V}_{11}\}, \quad (1)$$

where $\mathfrak{V}_i$ (i = $n_{\mathfrak{V}}$, $n_{\mathfrak{V}}$ is the number of characteristics) are subsets with elements that characterize the features of the system architecture; $\mathfrak{v}_{10}$ is an element that determines the presence of a controller in the system; set $\mathfrak{V}_{10,1}$ is a one-element set; $\mathfrak{v}_1, \mathfrak{v}_2, \dots, \mathfrak{v}_{11}$ are the designations of elements in sets $\mathfrak{V}_1, \mathfrak{V}_2, \dots, \mathfrak{V}_{11}$, respectively.

Thus, the number of systems of type $\mathfrak{S}$ according to the $\mathfrak{P}$ principle is different, but according to formula (1), they are all united by the presence of a controller in their architecture. The number of subsets $\mathfrak{V}_i$ (i = $n_{\mathfrak{V}}$, $n_{\mathfrak{V}}$ is the number of characteristics) can be different, including less than $n_{\mathfrak{V}}$, but the presence of the one-element set $\mathfrak{V}_{10,1}$ and the set $\mathfrak{V}_{11}$ in the direct product of sets is mandatory.

Such a division of the system architecture by internal structure makes it possible to determine the necessary elements and components in the system architecture, which will contain a controller and specialized functionality, and is the basis for developing the concept and methodological foundations for the synthesis of such systems. In contrast to the known principles of the synthesis of multi-computer systems with combined baits and traps and a decision-making controller for detecting and countering malware and attacks, the pro-

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

posed principle of the synthesis of such systems contains two defining requirements for the system architecture. The decision-making controller is separated from the system center, which makes it possible to form its architecture separately from the architecture of the system center and, as a result, to make decisions on the decisions developed in the system center independently of it. This is due to the specifics of the system and gives the system an advantage over attackers or their tools, as it generates different final system responses under the same initial conditions at different times, which confuses attackers.

Let us consider a method for synthesizing systems with partial centralization in their architecture. Simultaneously, the synthesis method does not define a controller, but primarily implements partial centralization and investigates the degree of degradation of such systems depending on the time of their operation and the impact of malware and computer attacks.

4. Method for creating partially centralized systems for detecting malware in computer networks

A method for creating partially centralized systems to detect malware in computer networks was developed. The generalized scheme is presented in Fig. 1.

In partially centralized distributed systems, it is necessary to synthesize the following principles of operation, functional features and characteristics: 1) formation of the system from components; 2) communication between system components; 3) maintaining the integrity of the system; 4) partial centralization; 5) migration of the decision-making center of the system; 6) assessment of the state of components and the system; 7) evaluation of the results of distributed calculations in components; 8) formation of a decision in several components; 9) reorganization of the system architecture; 10) determination of further steps of the system at the current time; 11) completion of the functioning of the components and the system.

Let's detail each of the given principles of functioning and characteristics. All of these must be synthesized in such systems completely. Because of such a synthesis, the system will become self-organized, adaptive, and partially centralized.

Formation of system S from components can be performed at the beginning of its installation and activation, during operation if necessary, and after turning on the computer stations in the network. In addition, new components can be added to the system or existing ones can be removed. In addition, some of the computer stations in which the components are installed may be turned off for a long time; therefore, the system will contain a smaller number of components.

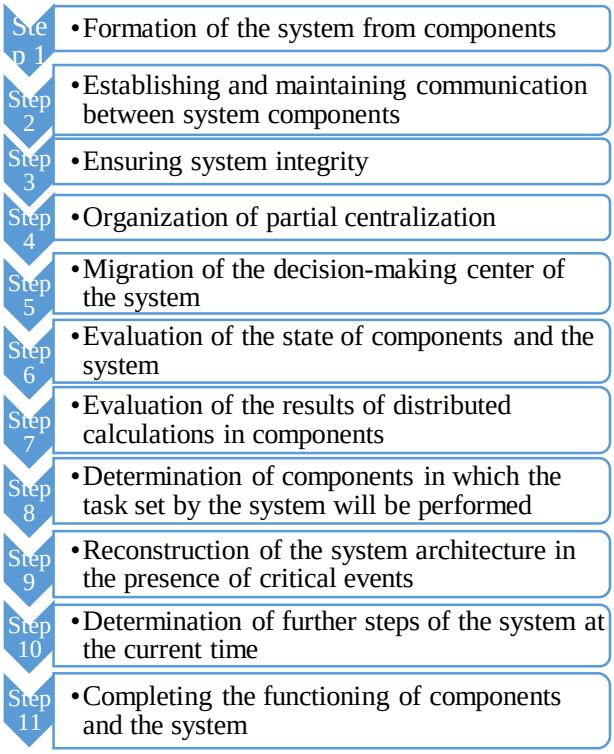

Fig. 1. Generalized scheme of the method

Computer stations with system components can be turned on at the same time or at different times. Computer stations may not be turned off, that is, they may be turned on all the time. These cases will influence the formation of system S. Let's set them in the system so that its decision-making center can consider these cases and their variations in the process of system formation and functioning, and as an active last event. Variants of the formation of the system are defined as a set,

$$M_S^{var,1} = \left\{ m_{S,1}^{var,1}, m_{S,2}^{var,1}, \cdots, m_{S,n_{M_S^{var,1}}}^{var,1} \right\},$$

where $n_{M_S^{var,1}}$ is the number of options. For example, the following elements: $m_{S,1}^{var,1}$ – characterizes the formation of the system at its beginning and activation; $m_{S,2}^{var,1}$ - characterizes the formation of the system in the process of functioning as needed; $m_{S,3}^{var,1}$ – characterizes the formation of the system after turning on computer stations in the network. The options that are set by the plural $M_S^{var,1}$, there can be only one at the current moment of time. That is, system S will analyze the last version of its formation. To determine the last variant of system formation, we introduce a predicate on the elements of the set as follows: $M_S^{var,1}$. The options that are

set by the plural, there can be only one at the current moment of time. That is, the system analyzes the last version of its formation. To determine the last variant of system formation, we introduce a predicate on the elements of the set as follows: The options that are set by the plural, there can be only one at the current moment of time. That is, the system analyzes the last version of its formation. To determine the last variant of system formation, we introduce a predicate on the elements of the set as follows:

$$P_S^{var,1}\left(m_{S,q}^{var,1}\right)=\begin{cases}0, m_{S,q}^{var,1} & \text{— not current version,}\\ q, m_{S,q}^{var,1} & \text{— current version,}\end{cases}\quad q=1,2,\dots,n_{M_S^{var,1}}. \qquad (2)$$

Similarly, we introduce the set of variations by the set $M_S^{var,2}=\left\{m_{S,1}^{var,2}, m_{S,2}^{var,2}, \cdots, m_{S,n_{M_S^{var,2}}}^{var,2}\right\}$, here $n_{M_S^{var,2}}$ is the number of variations. For example, the following elements: $m_{S,1}^{var,2}$– supplementing the system with new components; $m_{S,2}^{var,2}$– removal of components from the system. For variations given by a set of cases $M_S^{var,2}$, we introduce a predicate, the value of which will reflect their presence or absence, so

$$P_S^{var,2}\left(m_{S,q}^{var,2}\right)=\begin{cases}0, m_{S,q}^{var,2} & \text{— not current version,}\\ q, m_{S,q}^{var,2} & \text{— current version,}\end{cases}\quad q=1,2,\dots,n_{M_S^{var,2}}. \qquad (3)$$

Similarly, we introduce the set of variations by the set $M_S^{var,3}=\left\{m_{S,1}^{var,3}, m_{S,2}^{var,3}, \cdots, m_{S,n_{M_S^{var,3}}}^{var,3}\right\}$, where $n_{M_S^{var,3}}$ is the number of variations. For example, the following elements: $m_{S,1}^{var,3}$– computer stations, in which there are system components, turned on at the same time; $m_{S,2}^{var,3}$ – computer stations in which there are system components that are turned on at different times; $m_{S,3}^{var,3}$ – computer stations, in which system components are present, are not turned off for the entire time of system operation. For variations given by a set of cases $M_S^{var,3}$, we introduce a predicate, the value of which will reflect their presence or absence:

$$P_S^{var,3}\left(m_{S,q}^{var,3}\right)=\begin{cases}0, m_{S,q}^{var,3} & \text{— not current version,}\\ q, m_{S,q}^{var,3} & \text{— current version,}\end{cases}\quad q=1,2,\dots,n_{M_S^{var,3}}. \qquad (4)$$

Similarly, we introduce the set of variations by the set

$$M_S^{var,4}=\left\{m_{S,1}^{var,4}, m_{S,2}^{var,4}, \cdots, m_{S,n_{M_S^{var,4}}}^{var,4}\right\},$$

where $n_{M_S^{var,4}}$ is the number of variations. For example, the following elements: $m_{S,1}^{var,4}$ – part of the computer stations in which the components are installed may be turned off for a long time, the system will contain a smaller part of the components, and at this time its current formation may occur caused by certain events without these components; $m_{S,2}^{var,4}$ – the new formation of the system did not occur without the components that were located in the switched off computer stations. For variations given by a set of cases $M_S^{var,4}$, we introduce a predicate, the value of which will reflect their presence or absence:

$$P_S^{var,4}\left(m_{S,q}^{var,4}\right)=\begin{cases}0, m_{S,q}^{var,4} & \text{— not current version,}\\ q, m_{S,q}^{var,4} & \text{— current version,}\end{cases}\quad q=1,2,\dots,n_{M_S^{var,4}}. \qquad (5)$$

Formulas (2) – (5) describe the stage at which system S is formed and specify its variants. The results of the predicate calculation form part of the input data for the system's decision-making center. After installing all the components of the system in the computer stations in the network, considering the components with and without the decision center, when the system is first started, the components with the decision center check the predicate values for the various elements of the set $M_S^{var,1}$ and establish that all values are equal to zero. Then, the system will independently, without a user or administrator, begin the initial formation of its components from the existing subset functions, and after the completion of such formation, it will proceed to the division of components with a decision-making center into active and inactive ones.

To ensure communication between components in system S, we will organize communication between components not only using the standard sending of messages with the appropriate number of confirmation messages, but also with the sequential addition of certain tasks to them, the result of which is known in the components that plan to send the main message or task, as well as conducting an analysis of the time spent between sending the first connection request and receiving the results of the test task. In general, the entire system S will act as one big sensor that will respond to changes in the operation of its parts, including communication between components. If all the components are turned off at the same time, then they could fix a certain task in themselves, the execution of which they should perform after the next turn on. Shutting down computer stations may be correct, and then such an action of fixing the

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

same control task for its use as confirmation of the legitimacy of the connection with the corresponding component could be implemented and fixed statically. However, it may happen that the computer station will turn off in an emergency, and such fixation of a certain control task will not happen. Thus, the introduction of redundancy in the organization of communication between components requires consideration of options with enabled and disabled computer stations and synchronization of the time during which the components are active and establish communication with each other. Therefore, let's introduce a set of options

$$M_S^{var,5} = \left\{ m_{S,1}^{var,5}, m_{S,2}^{var,5}, \cdots, m_{S,n_{M_S^{var,5}}}^{var,5} \right\},$$

where $n_{M_S^{var,5}}$ is the number of options that arise when redundancy is introduced to organize communication between components. The elements of the set are as follows: $m_{S,1}^{var,5}$– computer stations in which system components are available, turned on at the same time; $m_{S,2}^{var,5}$ – computer stations, in which system components are present, are turned on at different times, and a part may be turned off after a certain time of operation, and a certain part may be turned on after this time, or not turn on at all for a long time. Accordingly, the system S components should also be active only when the computer stations are turned on and functioning. Let's also introduce a set of options

$$M_S^{var,6} = \left\{ m_{S,1}^{var,6}, m_{S,2}^{var,6}, \cdots, m_{S,n_{M_S^{var,6}}}^{var,6} \right\},$$

where $n_{M_S^{var,6}}$ is the number of options that arise when the computer stations, in which the system components are installed, are terminated. The elements of the set are as follows: $m_{S,1}^{var,6}$ – computer stations, in which the system components are present, turned off correctly at the same time; $m_{S,2}^{var,6}$ – computer stations, in which there are system components, turned off by emergency at the same time; $m_{S,3}^{var,6}$ – computer stations that have system components, turned off at different times correctly; $m_{S,4}^{var,6}$– computer stations, in which there are system components, turned off at different times, partly correctly and partly in an emergency. According to the given sets, it is possible to form two-element sets that characterize the events related to communication in the system depending on the computer stations as follows:

$$\{m_{S,1}^{var,5}; m_{S,1}^{var,6}\}; \{m_{S,1}^{var,5}; m_{S,2}^{var,6}\}; \{m_{S,1}^{var,5}; m_{S,3}^{var,6}\}; \{m_{S,1}^{var,5}; m_{S,4}^{var,6}\}; \{m_{S,2}^{var,5}; m_{S,1}^{var,6}\}; \{m_{S,2}^{var,5}; m_{S,2}^{var,6}\}; \{m_{S,2}^{var,5}; m_{S,3}^{var,6}\}; \{m_{S,2}^{var,5}; m_{S,4}^{var,6}\}.$$

For individual computer stations, it is necessary to develop similar tasks in sets because, according to them, communication between individual components in the system will be ensured. In general, in the system, communication between components and sending messages will be established according to the following relations: "one to all" ($m_{S,1}^{var,7}$); "all to one" ($m_{S,2}^{var,7}$); "to each other" ($m_{S,3}^{var,7}$); "one to a certain number, but not to all" ($m_{S,4}^{var,7}$); "a certain number, but not all, to one" ($m_{S,5}^{var,7}$); "a certain number, but not all, to a certain number, but not to all" ($m_{S,6}^{var,7}$). Let's define these relations as a set

$$M_S^{var,7} = \left\{ m_{S,1}^{var,7}, m_{S,2}^{var,7}, \cdots, m_{S,n_{M_S^{var,7}}}^{var,7} \right\},$$

where $n_{M_S^{var,7}}$ is the number of and $n_{M_S^{var,7}} = 6$.

To specify the connection between individual computer stations, we introduce a set of options

$$M_S^{var,8} = \left\{ m_{S,1}^{var,8}, m_{S,2}^{var,8}, \cdots, m_{S,n_{M_S^{var,8}}}^{var,8} \right\},$$

where $n_{M_S^{var,8}}$ is the number of options that arise in the process of establishing a connection between the computer stations in which the system components are installed. The elements of the set are as follows: $m_{S,1}^{var,8}$ – a computer station in which the system component is present, turned on; $m_{S,2}^{var,8}$ – the computer station, in which the system component is present, is turned off correctly; $m_{S,3}^{var,8}$ – a computer station, in which a system component is present, is turned off by emergency. According to the given set, we will form two-element subsets that characterize the state of the computer stations regarding their start and end of work as follows: $\{m_{S,1}^{var,8}; m_{S,2}^{var,8}\}; \{m_{S,1}^{var,8}; m_{S,3}^{var,8}\}$. Therefore, if the state of the computer station, in which the system S component is present, is characterized by the subset $\{m_{S,1}^{var,8}; m_{S,2}^{var,8}\}$, then the messages it receives and sends will be considered by the decision-making center to be executed correctly. Otherwise, that is, for a subset $\{m_{S,1}^{var,8}; m_{S,3}^{var,8}\}$, the decision-making center records such an event and, when the computer station is turned on next, processes an additional special procedure for establishing communication with this component to update this component in the system. In addition, when performing a standard communication action between any two components of the system, regardless of the type of element of the set $M_S^{var,7}$, the performance of an

additional check is mandatory and consists of the performance of a certain task of the component that plans to establish a connection and an equally certain task from the component with which communication is planned.

Thus, the establishment of communication between system components in different nodes in the network will be carried out considering the types of relations that allow the synthesis of partial centralization and additional verification of the legitimacy of the component.

An enterprise's corporate network can have several segments. Components of system S can be installed in different parts of the network and remotely in home computer stations. Within the corporate network, switches may fail or there may be other reasons that will cause the system to be divided into two or more unrelated subsystems. That is, the system in the process of functioning can disintegrate into unrelated parts. Then, each of the parts transforms itself into a reduced system S and continues to work if at least two active components with a decision-making center remain in each of the parts. If one of the parts does not have active components with a decision center and is inactive, then the components of this part block the operation of the computer stations and issue a corresponding message to the administrator. If components with a decision center are inactive at the moment of a certain emergency or intentional separation of the second part, which will contain all active components with a decision center of the system, then their transfer to the active state will occur after another communication session and establish the absence of active components with the center decision-making. Maintaining the integrity of system S during its operation will be ensured by the periodic exchange of messages between the system components according to relations from the set $M_S^{var,7}$, which will be chosen randomly. In addition to these two cases, which characterize ensuring the integrity of the system, there is also a case related to the partial centralization synthesis in system S. If part of the active components, which contain the decision-making center of the system, is removed from the system for certain reasons, then the remaining part will begin the procedure of forming the system from the existing components. However, if there are less than two such components, then all active components, including those without the functionality with a decision-making center, will block the operation of computer stations and will issue a corresponding message to the system administrator. Thus, the given organization of system integrity support considers the possibility of synthesis in system S of partial centralization and adaptability.

The partial centralization of the system is specified in its designed architecture, in particular by the following formula (4, [1]). The system is partially centralized because all its components are divided into two subsets: a subset of components that can be the center of the system and a subset of components that lack functions to ensure the functioning of the decision-making center of the system. The management of the entire system occurs from the components in which the decision-making center of the system is located. Therefore, it is centralized. Partial centralization is ensured by the fact that the components of system S, in which the decision-making center of system S for decision-making is located, develop proposals separately in each of these components, that is, decentralized, and agree to it jointly by all. Thus, the system is not fully centralized.

We will consider partial centralization in relation to the components that may contain the decision-making center of the system. Most of the installed components of system S in computer stations must contain functionality that ensures the functioning of the decision-making center of the system. After the installation of the system is completed, the system is started for the first time with all the computer stations in which the system components are installed turned on. At this stage of the system's functioning, all components that may have a decision-making center of the system will participate in the preparation of the first final decision to determine the first step of the system. This solution will reduce the number of active components of the decision-making center by switching some of them to an inactive state. Let's set the set of states into which system S can go

$$M_S^{st} = \left\{ m_{S,1}^{st}, m_{S,2}^{st}, \cdots, m_{S,n_{M_S^{st}}}^{st} \right\},$$

where $n_{M_S^{st}}$ is the number of states. Then, $m_{S,1}^{st}$ – the state of the system in which the active components of the decision-making center are updated. The decision to transition to this state is determined by the active components of the system's decision-making center. Implementation of system management is determined by the decision-making center of the system. Decisions will be formed and instructions will be sent to the components for their implementation. The formation of the decision in the system will be carried out in the active components of the decision-making center. If we consider them collectively with the number of more than one, at the architectural level, they can be positioned as a decentralized subsystem. Therefore, the formation of the final solution will be performed according to the solutions that will be obtained from the active components and their processing. The completion of the process of working out the final solution will be the transition of the system to a state. The transfer of the decision from the active components of the decision-making center of the system to the specified components will be per-

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

formed according to the relation set by one of the elements $m_{S,5}^{var,7}$ or $m_{S,6}^{var,7}$, and will transfer the system to the next state. Thus, the main steps in the synthesis of partial centralization in the system are to ensure the implementation of the formation of components in which the decision-making center will function, the formation of decisions in the components and the final decision, and the processing of the final decision in terms of sending it to the components in which it should be done. To form decisions, the corresponding components receive certain messages or results in the process of system operation.

The elements of the set of states M_S^{st} will specify the current states of the system as a whole and its components. Transitions from state to state are set in ordered pairs $(m_{S,p}^{st}, m_{S,q}^{st})$, where p is the number of the current state of the system and q is the number of the next state of the system. Both states of the system will necessarily apply to its component. In particular, the transition from the current state to the next state may not cover all components of the system in terms of performing certain actions to achieve a complete transition of the entire system to the next state. Thus, the set states M_S^{st} will characterize the system as a whole, and for the system components M_S^{st}, they will be the same according to the list of elements of the set, but the state of individual components will be specified separately, because the components may not be in the same state at the same time. Individual components of the system can change their state according to the elements of the set M_S^{st} more often than the system. When the system transitions from state to state, a certain part of the components may be involved in the process, and their states may also change as a result. It is assumed that the functioning of the system is possible in the presence of at least two components, which can be the center of the functioning of the system. Thus, scaling the system through its states for a minimal number of components is admissible. The transition from state to state is provided by a certain set of functions. In one cycle, the system can change several states if it decides to do so. The system can form new states by combining the states known to it. In a certain state, the system receives current and input data, for the processing of which appropriate functions will be involved. As a result, a field of events for processing is formed, which is defined by the set of events

$$M_S^{pd} = \left\{ m_{S,1}^{pd}, m_{S,2}^{pd}, \dots, m_{S,n_{M_S^{st}}}^{pd} \right\},$$

where M_S^{pd} is the number of events.

When the number of enabled computer stations changes, the number of components in the system changes, particularly those that may have a decision-making center in the system. Moreover, during a certain time of the system's operation, events may occur that will require a change in the state of the system in relation to some components that may contain the system's decision-making center. Therefore, the migration of the decision-making center of the system between certain components must be specified by certain appropriate functions for its implementation by the system itself.

Considering the target orientation of system S for the detection of malicious software, it is necessary to determine, in addition to the current state of the components and the system, the security state of the computer stations in which the components are installed and their own security state. Thus, to ensure the proper functioning of the system and to make decisions regarding its further functioning, the following states need to be taken into account: the state of the system, the states of components, and the states of computer stations. The values of these states will be determined not only with respect to their safety in relation to the effects of malicious software but also with respect to the general loading of computer station resources and the load of executed tasks in the component. We integrate the general states of the components and computer stations into one system component state indicator according to formula (53, [1]), according to which we calculate the value for each component $a'_{3,S_{1,n}}$. We determine the state of the system as a whole according to the states of its components, considering the values $a'_{3,S_{1,n}}$ for all components that are currently active in the system, as follows:

$$a_{S,t}^{st,1} = \frac{1}{p} \cdot \sum_{q=1}^{p} a'_{3,S_{i,n},q}, \qquad (6)$$

where p is the number of active system components in enabled computer stations; $p = 1,2,\dots,n$, n – the number of components in system S; $a'_{3,S_{I,n},q}$ – the value $a'_{3,S_{1,n}}$ in the q ith component.

In the components of system S, calculations will be carried out and transferred to the active components, in which the decision-making center of the system will function. In the active components of the decision-making center, certain tasks can also be performed and their results obtained. Under certain circumstances, not all components can receive the result of the assigned task and transmit it within the given time intervals. In addition, in certain components, the results of the performance of the assigned task may be different from the results obtained from most components that were involved in its performance. Not all the results of the assigned task will have clear expected numerical values. For the formation of the final result for its use in determining the further steps of the system in the components of the decision-making center, it is necessary to divide the components from which the results of the

task were obtained into two classes. All tasks that can be performed by the system are divided into subset functions that can perform them and types of components in which they can be performed. The states of the components will constantly change. They do not have static numerical values. The values of the characteristic indicators of the system components, depending on the types of tasks performed are determined by formulas (12, [1]), (44, [1]) and (53, [1]). Calculate each value a'_{1,S_i}, $a'_{2,S_{k+1,n}}$, $a'_{3,S_{1,n}}$ for individual components of the system, functions with five arguments $f_{a'_{1,S_i}}$, $f_{a'_{2,S_{k+1,n}}}$, $f_{a'_{3,S_{1,n}}}$ are used, respectively. For task types, only one of the three values will be calculated. But its value will be obtained according to the five arguments of the corresponding function. Let's consider options for defining the functions $f_{a'_{1,S_i}}$, $f_{a'_{2,S_{k+1,n}}}$, $f_{a'_{3,S_{1,n}}}$.

The first option can be used to determine the values of the functions $f_{a'_{1,S_i}}$, $f_{a'_{2,S_{k+1,n}}}$, $f_{a'_{3,S_{1,n}}}$ is given by the arithmetic mean value of all five arguments as follows:

$$A'_{1,S_I} = f_{a'_{1,S_I}}\left(a'_{1,S_I,1}, a'_{1,S_I,2}, a'_{1,S_I,3}, a'_{1.S_I,4}, a'_{1,S_I,5}\right) = \cdot = \frac{1}{5} \cdot \sum_{q=1}^{5} a'_{1,S_I,q}, \quad (7)$$

$$a'_{2,S_{k+1,n}} = f_{a'_{2,S_{k+1,n}}}(a'_{2,S_{k+1,n},1}, a'_{2,S_{k+1,n},2}, a'_{2,S_{k+1,n},3}, a'_{2,S_{k+1,n},4}, a'_{2,S_{k+1,n},5}) = \frac{1}{5} \cdot \sum_{q=1}^{5} a'_{2,S_{k+1,n},q}, \quad (8)$$

$$A'_{3,S_{I,n}} = f_{a'_{3,S_{I,n}}}\left(a'_{3,S_{I,n},1}, a'_{3,S_{I,n},2}, a'_{3,S_{I,n},3}, a'_{3,S_{I,n},4}, a'_{3,S_{I,n},5}\right) = \cdot = \frac{1}{5} \cdot \sum_{q=1}^{5} a'_{3,S_{I,n},q}, \quad (9)$$

After obtaining values according to formulas (7) – (9) for each of the components to which the task was sent for execution, it is necessary to divide these values into two classes. The first class will include those values that are equal to or are closest to one on the numerical axis, and the rest will be included in the second class. Then, the results of the task, which are obtained from components with values from the first class, will be accepted with the appropriate degree of confidence. If they are numerical, the arithmetic mean value will be calculated as the final result. If the values of the performed task are non-numeric, then the result of the execution will be accepted as completed if the first class is not empty. If the first class is empty, the task is performed again. To form two classes, we form an interval for the values (a'_{1,S_i}, $a'_{2,S_{k+1,n}}$, $a'_{3,S_{1,n}}$) from the components so, so that the minimum of them is the lower limit of the interval, and the upper limit of the interval is the number one. The interval formed in this way will constantly change for each new task, since the lower limit will be changed. Let's set the lower limit for the first class as 20% of the deviation from unity to the lower limit, and for the second class, respectively, as 80% of the deviation from the lower limit of the interval. The common value of the two classes will be assigned to the first class, then the values of the second class will be in the interval with an open upper border. Let's define an interval with classes as follows: and for the second class, respectively, as 80% deviation from the lower limit of the interval. The common value of the two classes will be assigned to the first class, then the values of the second class will be in the interval with an open upper border. Let's define an interval with classes as follows: and for the second class, respectively, as 80% deviation from the lower limit of the interval. The common value of the two classes will be assigned to the first class, then the values of the second class will be in the interval with an open upper border. Let's define an interval with classes as follows:

$$A'_{1,S} = \min\left(a'_{1,S_1}, a'_{1,S_2}, \dots, a'_{1,S_p}\right); p \le I;$$
$$a'_{1,S} = \min\left(a'_{2,S_{k+1}}, a'_{2,S_{k+2}}, \dots, a'_{2,S_p}\right); p \le n;$$
$$a'_{1,S} = \min\left(a'_{3,S_1}, a'_{3,S_2}, \dots, a'_{3,S_p}\right); 1 \le p \le n, \quad (10)$$

where $[a'_{1,S}; 1]$ – the range of all values; $a'_{2,S} = 1 - 0{,}2 * (1 - a'_{2,S}; 1)$ – the limit value of both classes; $[a'_{2,S}; 1]$ – range for values from the first class; $[a'_{1,S}; a'_{2,S}]$ – range for values from the second class

Thus, the clustering of values performed according to formula (10) allows the decision-making center of the system to accept the results of the task in the given components.

However, when evaluating the results of distributed calculations in components according to the first option, the weight of the value of a certain characteristic indicator is leveled and can affect the assignment to a certain class. This is because, according to formulas (7) – (9), all terms are considered equivalent, regardless of their weight. Considering their weights in the overall resulting value is complicated, because these weights do not have established values and require the involvement of experts to determine them, which will affect the reduction of the self- organization of the system and their possible accuracy, in connection with the constant changes of states in computer stations. Therefore, consider the second option for determining the values of the functions $f_{a'_{1,S_i}}$, $f_{a'_{2,S_{k+1,n}}}$, $f_{a'_{3,S_{1,n}}}$.

To perform clustering into two classes according to the second option, we consider a five-dimensional

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

space in which the five arguments of the functions $f_{a'_{1,S_i}}$, $f_{a'_{2,S_{k+1,n}}}$, $f_{a'_{3,S_{1,n}}}$ will set the points. Thus, when obtaining the values of the arguments of the functions from the system components in which the task was performed, the coordinates of points in the five-dimensional space will be formed from them, with the subsequent division of the points into two classes. The choice of the classification algorithm and the metric will be made based on the fact that the valuable values for the system will be those that will be closest to the value equal to one. Accordingly, it is necessary to choose a classification algorithm and metric in such a way that the first class is formed, in which the element (1;1;1;1;1) would be, or the cluster would be formed in its absence, but with coverage of the area of the points closest to it.

Let's consider the known metrics and select a metric for use in classification. The Euclidean distance metric specifies the geometric distance between objects in space. The squared Euclidean distance metric is characterized by giving more weight to the most distant objects. The Manhattan distance metric reduces the impact of individual long distances. The power-law distance metric is used when it is necessary to increase or decrease the weight for the dimensions of objects that differ significantly. Its disadvantage is the need to set two parameters. The Chebyshev metric is used if two objects differ by at least one coordinate. From the analyzed metrics, we will choose the Chebyshev metric, since according to it, it is possible to distinguish between two objects that differ by one coordinate, because the rest of the metrics with several different numerical coordinates can lead to certain identical distance calculations, which is inadmissible for the construction of the second variant of clustering. The Chebyshev metric defines the distance as follows:

$$\rho(x, x') = \max_{q=1,2,\dots,5}\left(\left|x_q - x'_q\right|\right), \qquad (11)$$

where x'_q is the coordinate of the center of the cluster; $q = 1,2,\dots,5$; x_q is the coordinate of a point in space.

For clustering, we will use the k-means method because, according to its application results, all objects will be divided into relatively homogeneous classes. Achieving division into classes is ensured by minimizing the sum of squared distances between each of the five values of the characteristic indicators, i.e., the arguments of the functions $f_{a'_{1,S_i}}$, $f_{a'_{2,S_{k+1,n}}}$, $f_{a'_{3,S_{1,n}}}$ and the center of the cluster, which is set as follows:

$$d_v = \sum_{i=p_1}^{p_2}\left(\max_{q=1,2,\dots,5}\left(\left|\alpha'_{w,S_i,1,q} - x'_q\right|\right)\right)^2, \qquad (12)$$

where x'_q is the coordinate of the center of the cluster; $q = 1,2,\dots,5$; for the value function $f_{a'_{1,S_I}}$, $p_1 = 1$, p_2 is the number of active components with the decision-making center of the system; for the function $f_{a'_{2,S_{k+1,n}}}$ value $p_1 = k + 1$, p_2 is the number of active components without functionality for the decision-making center of the system and $p_2 \le n$; for the function $f_{a'_{3,S_{1,n}}}$, value $p_1 = 1$, p_2 is the number of active components and $p_2 \le n$.

At a certain step of the iteration, the value of the element specified by five coordinates will be chosen as the center of the cluster. We quantitatively establish two clusters for separating values. We record the values obtained from the components in which the task was performed in all active components that form the decision-making center of the system. Let us take as the center of the first cluster the value given by the coordinates (1;1;1;1;1), and as the center of the second cluster, the value of the characteristic indicator, which is the most distant from the point with coordinates (1;1;1;1;1). If there are several such values, we take the last considered value that is suitable as the center of the cluster. The rest of the values are distributed between two classes according to formula (12), depending on the distance to the two centers of the two clusters in such a way that the class includes the value with the smallest distance according to the Chebyshev metric (formula (11)). To achieve the stability of clusters, that is, to assign the same values to the clusters, the centers of the clusters need to be clarified through repeated iterative calculations. To select the next center of the cluster, we find the arithmetic mean value of all the values of the characteristic indicators that are part of a certain cluster. The search for such centers is carried out until the same values that were in the previous step of the iteration at another cluster center remain in the clusters. As a result, it is achieved that the variance between classes will be maximized and that between elements will be minimized. To clarify the center of the cluster in the active components of the decision-making center, it is necessary to organize iterative steps. In the future, at the next steps of the tasks, these clusters will be needed at the next stages of the same task to evaluate the discrepancy. In addition, in the presence of previous stories from the performance of the same assigned task, the decision-making center of the system will average the values of the class limits based on the results of previous calculations to avoid system degradation, correct the assessment of task performance in the system, and fix the result of the completed task. Thus, the classification according to the k-means method is divided into two classes according to the second variant of the values of the characteristic indicators of the active components and

the functions $f_{a'_{1,S_i}}$, $f_{a'_{2,S_{k+1,n}}}$, $f_{a'_{3,S_{1,n}}}$, as implementing the calculation of these values according to the Chebyshev metric.

To determine the components in which the task set by the system will be performed, the decision-making center of the system determines their security level through a survey of all components, which is calculated according to formula (6). Then, the center of decision-making according to the first variant of division into classes (formulas (7) – (9)) determines half of the components, i.e., a factor of 0.5 is put into formula (9) instead of 0.2, in which the given task will be performed. If the assigned task requires immediate execution or has a status related to security research in components, it is performed by all components. A smaller number of components may be involved in the performance of the task, if there are many of them, but this number cannot be less than ten components. This is because of the need to have a sufficient sample of values to correctly determine the final result. At the same time, a certain part may not have enough time to complete it in the set time. If the number of components in the system is small, e.g., less than ten, all components are involved in the performance of the assigned task. Each of the function sets and functions subsets in the components and the system as a whole have priorities that affect the number of components involved in performing the assigned tasks. These functions in components have clear connections with the tasks for which they are intended, which affect the number of components involved in performing the assigned tasks. These functions in components have clear connections with the tasks for which they are intended, which affect the number of components involved in performing the assigned tasks. These functions in the components have clear connections with the tasks for which they are intended.

The formation of a decision-making center can be carried out quantitatively from two to all components in which the corresponding functionality is installed. Decisions about the number of active components of the decision-making center are made from the moment the system is started by all components in which the system's decision-making center is present. If active components with a decision center stop working during system operation and the system continues, then the decision center adds new components to maintain the number of such components. To do this, he transfers them to the active state. The decision on the number of active components is made randomly by each active component, and then their arithmetic mean value is found and its fractional part is discarded.

In the process of functioning of the system, information is accumulated in its components, which can be the decision-making center of the system. This information is necessary to consider when making subsequent decisions about the next steps. However, not all components will have the same information about the passed system states, so mechanisms and functions must be introduced into them, which will allow it to be updated to a certain level. Such information, which needs to be saved for use in determining the next steps of the system and which applies exclusively to ensuring the functioning of the system, includes: information about the number of components in the system over time since the start of its operation and the state of their activity or non-activity; information about all the tasks performed in the system and the components involved for this, as well as the decisions that were made and the primary results for their adoption. To update the current information in all components of the decision center, you need to perform a task, because of which the database of information on the latest events in the system will be updated and sent to all components of the decision center that are located in enabled computer stations. For components of the decision center that are located in non-enabled computer stations, such information will be updated the next time they are enabled. Saving such information will enable the system administrator to analyze and find the reason for stopping the system or computer stations by its components, to make decisions about further steps by the system, and to optimize the performance of assigned tasks. To update the current information in all components of the decision center, you need to perform a task, because of which the database of information on the latest events in the system will be updated and sent to all components of the decision center that are located in enabled computer stations. For components of the decision center that are located in non-enabled computer stations, such information will be updated the next time they are enabled. Saving such information will enable the system administrator to analyze and find the reason for stopping the system or computer stations by its components, to make decisions about further steps by the system, and to optimize the performance of assigned tasks. To update the current information in all components of the decision center, you need to perform a task, because of which the database of information on the latest events in the system will be updated and sent to all components of the decision center that are located in enabled computer stations. For components of the decision center that are located in non-enabled computer stations, such information will be updated the next time they are enabled. Saving such information will enable the system administrator to analyze and find the reason for stopping the system or computer stations by its components, to make decisions about further steps by the system, and to perform optimization in the performance of assigned tasks. as a result, the database of information on the latest

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

events in the system will be updated and sent to all components of the decision-making center located in the enabled computer stations. For components of the decision center that are located in non-enabled computer stations, such information will be updated the next time they are enabled. Saving such information will enable the system administrator to analyze and find the reason for stopping the system or computer stations by its components, to make decisions about further steps by the system, and to perform optimization in the performance of assigned tasks. as a result the database of information on the latest events in the system will be updated and sent to all components of the decision-making center located in the enabled computer stations. For components of the decision center that are located in non-enabled computer stations, such information will be updated the next time they are enabled. Saving such information will enable the system administrator to analyze and find the reason for stopping the system or computer stations by its components, to make decisions about further steps by the system, and to optimize the performance of assigned tasks. such information will be updated the next time they are turned on. Saving such information will enable the system administrator to analyze and find the reason for stopping the system or computer stations by its components, to make decisions about further steps by the system, and to optimize the performance of assigned tasks. such information will be updated the next time they are turned on. Saving such information will enable the system administrator to analyze and find the reason for stopping the system or computer stations by its components, to make decisions about further steps by the system, and to optimize the performance of assigned tasks.

Reconstruction of the system architecture may also be necessary in the case of the detection of anomalous events or malicious manifestations in the computer network or stations. In this case, some of the system components can turn off the computer stations and inform the decision-making center of the system about withdrawal from the system. Such events can be partially detected by components with existing functionality because of the establishment of communication between components at the beginning of work after turning on computer stations or when problems with the functioning of components are established in a certain computer station. Events of this type are processed by appropriate functions-subsets and the decision-making center performs management actions to rebuild the system architecture according to the element from the set of events M_S^{pd}.

The determination of further system steps and the transition to them at the current time depends on the events in the system, which are set by the set M_S^{pd}, the results of event processing by the functions of the system components, the set of options for steps, which are set by the set of states M_S^{st}, the results of the decision-making center of the system; and the possibility of performing the specified transition to the next state at the current time, since changes may have occurred in the system during preparatory measures; and the immediate execution of the transition with verification of its complete completion.

Events in system S will be processed by certain functions. If events occur at computer stations and in the network, the system may process them if they are visible to its sensors. System S must control all objects and processes that can be assessed as anomalies or malicious influences in the future. To achieve this, it must have sufficient sensors and functions to process the results. If there are not enough, the system may not be able to detect, for example, malware in computer networks. In addition, events can occur within the system itself. They can be caused by both external and internal influences. However, we will consider all events to occur in computer stations and networks and should be processed without division into types. The division into types of such influences and manifestations will be used in the development of methods for targeted analysis by type to identify anomalous manifestations and malicious manifestations caused by the types of relevant means. Events defined by the elements of a set M_S^{pd}, are systematised precisely through the characteristics of the elements. An increase in the number of elements of the set M_S^{pd} will require an increase in the number of functions in the system components. Events can also be specified by combinations of elements. In addition, events can simultaneously occur in different nodes in the network and be visible to system components.

The results of event processing by the functions of the system components will be used to determine further steps of the system by its decision-making center and, as a result, will lead to the appearance of new events. In general, the system will constantly monitor and process events. However, not all events will lead to a change of state or transition to the next state.

The set of options for steps, which are given by the set of states M_S^{st}, determines the ability of system S to perform the tasks that relate to the organization of its functioning in accordance with the principles of self-organization and adaptability. If the system S states are few, i.e., the elements of the set M_S^{st}, then the pairs of elements that will be used to set the options for steps will also be few. This provides an opportunity to ensure proper stability for system S. However, filling the components with function sets for solving specialized tasks, as well as the environment in computer stations, will be rapidly changing, so the number of elements of the set

M_S^{st} cannot be a small number. In this regard, the number of states can be a large number; therefore, the number of variants of steps and their combinations will also be quite large; as a result, it is impossible to describe them all unambiguously. To solve this problem, it is necessary to set the rules by which the system will form and determine the steps for further transition to the next state, i.e. the rules for selecting options from several formed steps. At the same time, a set of states M_S^{st} should be initially formed, the number of elements of which should be further increased by forming new states in the system as combinations of basic states. Such combinations are formed according to the combination of different states of all components of the system into a single state of the entire system. During its operation, each component changes its state. Thus, some possible combinations of basic states will add new elements to the set of states M_S^{st}. This will be done by the decision-making centre of the system. Let's define the following basic states of the system S, i.e. elements of the state set: M_S^{st}: $m_{S,1}^{st}$ – the state of the system, in which the active components of the decision-making center have been updated; $m_{S,2}^{st}$ – the state of the system, in which the evaluation of the state of the components and the system was carried out; $m_{S,3}^{st}$ – the state of the system, in which the communication between the system components is carried out; $m_{S,4}^{st}$ – the state of the system, in which further steps of the system are determined at the current time; $m_{S,5}^{st}$ – the state of the system, in which the migration of the decision-making center of the system was carried out; $m_{S,6}^{st}$ – the state of the system, in which the restructuring of the system architecture was carried out; $m_{S,7}^{st}$ – the state of the system, in which the decision-making center is formed in several components; $m_{S,8}^{st}$ – the state of the system, in which the evaluation of the results of distributed calculations in components is carried out; $m_{S,9}^{st}$– the state of the system, in which the functioning of components and others has been completed. For components, the same states will also exist, but if, for example, the system updates the active components of the decision center, then in the components, the states can be as follows: the functionality for activating the decision-making center of the system in the component is disabled; the functionality for activating the decision-making center of the system in the component is activated; and the state of the component has not changed, i.e., a transition to the same state has occurred.

Transitions from state to state of system S are shown in Fig. 1. For example, the selected segment in the figure shows the transition from state $m_{S,7}^{st}$ to state or $m_{S,3}^{st}$ vice versa, depending on the coordinate of the transition vector in the ordered pair ($m_{S,7}^{st}$; $m_{S,3}^{st}$) or ($m_{S,3}^{st}$; $m_{S,7}^{st}$).

Thus, system S will be in a state shown in Fig. 2. The details of the effects and means of changing the state are shown in Fig. 3 with an indication of the connections that can be influenced.

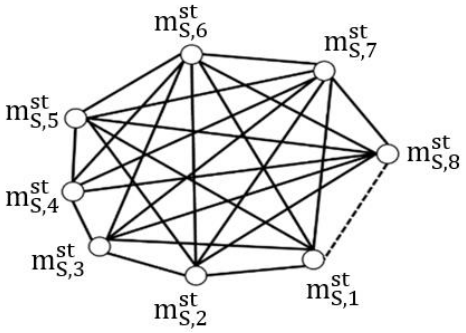

Fig. 2. System states and possible transitions between them

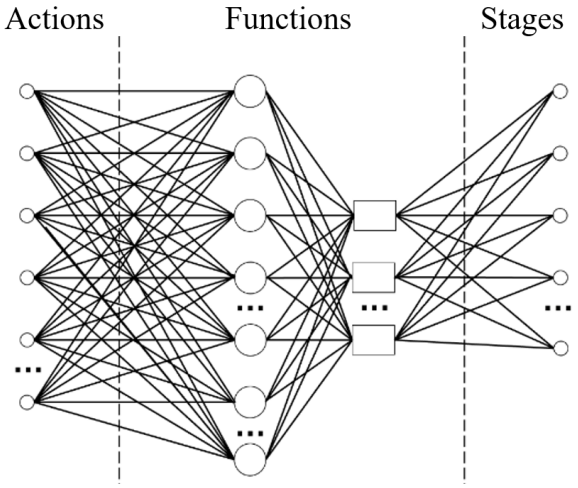

Fig. 3. Relationship between events, functions, and states

Two types of functions are highlighted in the depicted connection of events, functions, and states. The first type includes multiple functions in the components that do not belong to the functions of the decision-making center of the system, and the second type includes functions that form the decision-making center of the system. The highlighted segments between the two types of functions indicate that they refer to components that may be the center of the system's decision-making.

Depicted in Fig. 2 and 3 States refer exclusively to the system as a whole. The details of states in specific components are similar to the images in Fig. 1 and 2. System components can be in different states at the same time, and the state of the system is uniquely determined by the states of its components and the decision center components. A specific component of the

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

system can be in several states at the same time, which will be considered as a certain state formed by a combination of basic states. For example, system S adds components that became active as a result of turning on the computer stations, and the system component at this time evaluates the security level in the computer station. Then, these two states in the component will be combined into one at the current time, and the system will record the state of this component.

For the transition from state to state of system S, we will consider the activity of subset functions (matrix analysis from formula (6, [1])), the values of characteristic indicators (formulas (12, [1]), (44, [1]), (53, [1]), options for forming a system according to a set $M_S^{var,1}$ (formula (2)), and variations in forming a system according to sets $M_S^{var,2}$, $M_S^{var,3}$, $M_S^{var,4}$ (formulas (3) – (5)), introduction of redundancy in the organization of communication according to sets $M_S^{var,5}$ and $M_S^{var,6}$, type of relationship for establishing communication between components and sending messages according to the set $M_S^{var,7}$, specifying the connection of individual computer stations among themselves according to the set of options $M_S^{var,8}$, the set of events M_S^{pd}, the state of the system as a whole (formula (6)), the choice of options for calculating trust in the results of distributed calculations (formula (7) – (9) or according to clustering (formula (12))) and the set of states M_S^{st}. Let's set the next state of the system S through its current state and indicators of components and the system as follows:

$$m_{S,p}^{st} = F_{q\to p}^{S}\begin{pmatrix} m_{S,q}^{st} \\ M_S^{st} \\ M_S^{pd} \\ M_{S,k,I} \\ \alpha'_{1,S_i} \\ \alpha'_{2,S_{k+1,n}} \\ \alpha'_{3,S_{1,n}} \\ M_S^{var,1} \\ M_S^{var,2} \\ M_S^{var,3} \\ M_S^{var,4} \\ M_S^{var,5} \\ M_S^{var,6} \\ M_S^{var,7} \\ M_S^{var,8} \\ f_{\alpha'_{1,S_i}} \\ f_{\alpha'_{2,S_{k+1,n}}} \\ f_{\alpha'_{3,S_{1,n}}} \\ \alpha_{S,t}^{st,1} \end{pmatrix}, \qquad (13)$$

where $F_{q\to p}^{S}$ is a function that determines the next state of the system and sets the transition between the states.

When determining the next state of the system, there will be as many options as there are elements in the set M_S^{st}. The result of the selection can be the same state in which the system is already. In addition, the system can detect a state that is not in the set of states. This can happen when a combination of several states is established in a certain component or several components, which are set in the plural M_S^{st} by the basic constituent elements. Then, the system supplements this set of states with a new element formed by a combination of certain elements in certain components. However, the set of states is not formed in full from all the combinations at the beginning, but only from those that will appear during the functioning of the system.

Because the arguments of the function $F_{q\to p}^{S}$ are data of different types and the function must set rules according to which a discrete value will be determined, we set this function $F_{q\to p}^{S}$ as a general rule, which will contain a combination of the logical operators "AND" and "OR" and the negation "NOT" in the logical expression of local functions, which are assigned to each argument. Let be the $F_{q\to p,b}^{S}$ b-th local function, where $b = 1,2,\dots,19$, whose argument is the b-th argument of the function $F_{q\to p}^{S}$. The values of the local functions are discrete values {0} and {1}, where the value {0} will mean the fulfillment of the conditions for transition to the next state, and the value {0} will mean the fulfillment of such conditions. In a logical expression that forms a rule for defining a function, the values of local functions $F_{q\to p}^{S}$ can be combined with each other in full or in part, and can also form composite conditions, from which it is sufficient to move to a new state of fulfillment of one of the conditions.

For example, to change to a state $m_{S,1}^{st}$ in which the active decision center components are updated, you need to change the components that the system decision center will be in. The reasons for this change will be the updated data of such indicators and the results of the current state of the system indicators. If the previous state $m_{S,2}^{st}$ of the system , in which the assessment of the state of the components and the system is carried out, and it is established that the value $\alpha_{S,t}^{st,1} = 0.23$ is significantly less than the threshold value, then regardless of the remaining indicators, the system performs the functions necessary to update the active components of the decision-making center.

The given transitions from state to state will also maintain the integrity of the system and ensure its stability. Formula (13) defines the system S at the level of the states it can be in and the transitions between them,

which actually determines the processes that will function in it.

The decision-making center of the system according to formula (13) receives the result and establishes the possibility of carrying out the specified transition to the next state at the current time, since changes may have occurred in the system during preparatory measures.

The execution of the transition between the system states is provided by a subset function that checks its complete completion according to the communication specially specified in this case. If part of the system components did not have time to complete this transition, then in the future, when they are active, they reproduce the missed states in their history of states, renew the current indicators, and return to the current state of the system.

Completion of the functioning of the components and the system can be followed by their return to the performance of tasks when the computer stations are turned on, or when a given command is given to block part of the components or the system, or the removal of the components or the system as a whole from nodes in the network.

As a result, we set the main general steps of the method of organizing the functioning of partially centralized distributed systems according to the principles of self-organization and adaptability.

Step 1. Formation of the system S from components.

1.1. If system S is formed after the initial installation of all components (element $m_{S,1}^{var,1}$ of the characteristic set $M_S^{var,1}$, element $m_{S,1}^{var,3}$ of the set $M_S^{var,3}$), then each of its components receives information about the location of the remaining components in the computer network, records such information in its internal database and waits for the initial launch of one of the components by the administrator for subsequent initial launches of the remaining components after the indication from it about its start of operation.

1.2. If the computer stations are switched on constantly (element $m_{S,3}^{var,1}$ of the characteristic set $M_S^{var,1}$, element $m_{S,3}^{var,3}$ of the set $M_S^{var,3}$), then the formation of system S from components will be performed once and further changes (element $m_{S,2}^{var,1}$ of the characteristic set $M_S^{var,1}$) will be performed by the system itself when certain events occur.

1.3. If system S is formed after turning on computer stations in the network at the same time (element $m_{S,3}^{var,1}$ of the characteristic set $m_{S,3}^{var,1}$, element $m_{S,1}^{var,3}$ of the set $M_S^{var,3}$), then for its further functioning, all components perform a special procedure for exchanging messages to start functioning.

1.4. If the computer stations (element $m_{S,3}^{var,1}$ of the characteristic set $M_S^{var,1}$, element $m_{S,1}^{var,3}$ of the set $M_S^{var,3}$), in which the system components are installed, are turned on at different times, then the components that were in the first turned on computer stations form the system, and the rest are added to it after performing a special addition procedure of components in dynamic mode.

1.5. If new components are added to system S or existing components are removed (an element $m_{S,2}^{var,1}$ of the characteristic set $M_S^{var,1}$), then a special procedure for adding or removing components is used, followed by the formation of system S from existing active components that function in enabled computer stations. Supplementing system S with new components or removing existing components can be performed after substeps 1.1-1.4. The special procedure for adding and removing components involves the participation of the system administrator, the transition to the detail of substep 1.5 and the subsequent execution of substep 1.1.

1.5.1. Supplementing the system with new components (element $m_{S,1}^{var,2}$ of the set $M_S^{var,2}$, element $m_{S,1}^{var,3}$ of the set $M_S^{var,3}$, element $m_{S,2}^{var,4}$ of the characteristic set $M_S^{var,4}$) is performed when all computer stations in which the system S components are installed. Each component of system S is supplemented with information about new components, and new components are supplemented with information about all system components.

1.5.2. The removal of components from the system S(element $m_{S,2}^{var,2}$ of the set $M_S^{var,2}$, element $m_{S,2}^{var,3}$ of the set $M_S^{var,3}$) is performed using one computer station, in which the component containing the decision-making center of the system is installed. Through the component interface with administrator access rights, we provide an instruction to remove a specific component. Next, this system component sends a message about the withdrawal of the specified component to the rest of the system components that are active, i.e., they function in enabled computer stations. Components that will not be on computer stations that are turned on, i.e., will not receive this message about the removal of a specific component, but will receive this message when the computer stations in which they are installed are turned on, from the active components of the system's decision-making center.

1.5.3. When removing a single component at the current time (element $m_{S,2}^{var,3}$ set $M_S^{var,3}$) in which the decision center of the system is located, it is necessary to enable the computer station in which the component with the functionality of the decision center is present, or to use the passive component with the decision center at this current time. In this case, the passive component

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

or the attached component receives first an instruction about their sole control of the system, and then about the withdrawal of the given component.

1.5.4. The completion of sub-steps 1.5.1-1.5.3 is carried out by determining the last variant of system formation according to formula (2), according to which we calculate the predicate $P_S^{var,2}\left(m_{S,q}^{var,2}\right)$ $(q = 1,2,\dots,n_{M_S^{var,2}})$ on the elements of the set $M_S^{var,2}$.

1.5.5. After performing substep 1.5.4, we return to substep 1.5.

1.6. If the computer stations in which the system S components are installed (element $m_{S,2}^{var,1}$ of the characteristic set $M_S^{var,1}$, element $M_{S,1}^{var,4}$ of the characteristic set $M_S^{var,4}$) are not turned on for a long time, then the system is formed from the components that are in the switched on computer stations.

1.7. Step 1 is completed by determining the last variant of system formation according to formula (2) of substeps 1.2-1.4 and 1.6, according to which we calculate the predicate $P_S^{var,1}\left(m_{S,q}^{var,1}\right)$ $(q = 1,2,\dots,n_{M_S^{var,1}})$ on the elements of the set $M_S^{var,1}$, according to formula (4) of substeps 1.2-1.4 and 1.6, according to which we calculate the predicate $P_S^{var,3}\left(m_{S,q}^{var,3}\right)$ $(q = 1,2,\dots,n_{M_S^{var,3}})$ on the elements of the set $M_S^{var,3}$ and according to formula (5) of substeps 1.5.1 and 1.6, according to which we calculate the predicate $P_S^{var,4}\left(m_{S,q}^{var,4}\right)$ $(q = 1,2,\dots,n_{M_S^{var,4}})$ on the elements of the set $M_S^{var,4}$.

1.8. Depending on the events that will exclusively affect the formation of the system S architecture, and the results from substep 1.7, we return to one of substeps 1.2 - 1.4 or 1.6.

The results of substeps 1.5, 1.7, 1.8 are transmitted to the decision center of system S and processed by one of the defined substeps of the following steps.

The determination of the last variant of system formation according to formula (2) for the elements of the characteristic set $M_S^{var,1}$ and sub-steps 1.2-1.4 and 1.6 does not completely complete step 1, but only fixes the state of the system S after the complete execution of one of the sub-steps in certain time intervals, when no changes will occur in the system in its architecture. The execution of step 1 will be constant and independent of the rest of the steps because the architecture of the system may change constantly and will require the system S itself to react to such events through the execution of substeps of step 1.

We present the results of the substeps of step 1 in the table of their conjugation with the corresponding elements of the sets and the values of the predicates. As a result, we will receive information about the result of a certain substep and use it to make decisions about further steps of system S.

Step 2. Establish and maintain communication between system components.

2.1. If from the computer stations that have system components, at the current time when system S starts, there is only one computer station that is turned on (an element $m_{S,2}^{var,8}$ of the set $M_S^{var,8}$), then the system S component will use the "one to all" relationship after its loading (an element $m_{S,1}^{var,7}$ of the set $M_S^{var,7}$) , according to which a message will be sent to all system S components to establish communication with them.

2.2. If from the computer stations in which the system components are present, at the current moment of time at the current start of system S, all (element $m_{S,1}^{var,5}$ of the set $M_S^{var,5}$) are turned on and the decision-making center determines that the given component addresses all the remaining components, then the given system component will use the relation "one to all" (element $m_{S,1}^{var,7}$ of the set $M_S^{var,7}$), according to which a message will be sent to all components of the system S to maintain communication with the rest.

2.3. If one component is missing in system S because of the non-activation of the corresponding computer station, then all the remaining components periodically contact it to check its presence in order to form a complete system, that is, we perform the "all to one" relationship (element $m_{S,2}^{var,7}$ of the set $M_S^{var,7}$).

2.4. If a decision is made in system S to send a message to maintain and check the connection with a given component for certain reasons, then all the remaining components refer to it, i.e., we perform the "all-to-one" relation (element $m_{S,2}^{var,7}$ of the set $M_S^{var,7}$).

2.5. If a decision is made in the system S to send a message from a specific component to maintain and check communication with a given component for certain reasons, then we perform a "one-to-one" relationship (element $m_{S,3}^{var,7}$ of the set $M_S^{var,7}$).

2.6. If a decision is made in the system S to send a message from a specific component to maintain and check communication with a certain number of components, but not all, for certain reasons, then we perform the relation "one to a certain number, but not to all" (an element $m_{S,4}^{var,7}$ of the set $M_S^{var,7}$).

2.7. If a decision has been made in the system to send a message from a certain number of specified, but not all, components to one to support and verify communication with a given component for certain reasons, then we perform the relation "a certain number, but not all, to one" (element $m_{S,5}^{var,7}$ of the set $M_S^{var,7}$).

2.8. If a decision is made in the system to send a message from a certain number of specified, but not all, components to a certain number to support and check communication with them for certain reasons, then we perform the relation "a certain number, but not all, to a

certain number, but not to all" (element $m_{S,6}^{var,7}$ of the set $M_S^{var,7}$).

2.9. If a component is sent a message or an instruction, and it is currently turned off together with the computer station, then it sends a message to all components that are active, that is, those that are in the computer stations that are turned on, and we perform a one-to-one relationship to a certain number, but not to all" (an element $m_{S,4}^{var,7}$ of the set $M_S^{var,7}$) and commands or messages sent to her are nullified.

2.10. If a component is sent a message or instruction and it is currently shutting down along with the computer station, then it does not send a shutdown message to all components that are active, that is, those that are in the computer stations that are turned on. When the next time the computer station is turned on, the component notifies all other active components about the previous emergency event and communicates with them, performing a one-to-some, but not all, relationship (an element $m_{S,4}^{var,7}$ of the set $M_S^{var,7}$), but the commands or messages that were sent to it from certain components are canceled.

2.11. If when establishing a connection between system components, the standard part (according to the "flowering" scheme only to confirm the establishment of the connection and the activity of the components) and the additional part (according to the use of redundancy to additionally confirm the legitimacy of the components) were successfully completed for all system components in the computer stations that are turned on at the same time (an element $m_{S,1}^{var,5}$ of the set $M_S^{var,5}$), then system S will continue to function in regular mode.

2.12. If when establishing a connection between system components, the standard part (according to the "flowering" scheme only to confirm the establishment of the connection and the activity of the components) and the additional part (according to the use of redundancy to additionally confirm the legitimacy of the components) were successfully completed for all system components in the computer stations that are turned on at different times, and a part may be turned off after a certain time of operation, and a certain part may be turned on after this time or not turn on at all for a long certain time (an element $m_{S,2}^{var,5}$ of the set $M_S^{var,5}$), then the system S will continue to function in regular mode as part of active component in enabled computer stations.

2.13. If, for the cases of substeps 2.11, 2.12, when establishing a connection between system components, the standard part (according to the "flowering" scheme only to confirm the establishment of a connection and the activity of the components) is not completed successfully, then the system components that established such a fact about a certain component report about such a result to the decision-making center of the system.

2.13.1. If such a message is received from two components that have attempted to communicate with each other, then the decision center instructs them to retry the connection in the standard part of the procedure. In addition, it instructs another component to establish communication with these two components, and these three components must inform the decision-making center about the performance results.

2.13.2. If such a message is received from one of the two components that were attempting to communicate with each other, then the decision center instructs that component and the other two active components to attempt to establish communication according to the standard part of the procedure and inform these three components of the results and must inform the decision-making center.

2.14. If it is confirmed in substeps 2.13.1 and 2.13.2 that there are problems with establishing communication with a certain component according to the standard part of the communication establishment procedure, then such a component will be added to the list of components that need to be investigated by the decision-making center and will be periodically tested for connection with a certain number of components (an element $m_{S,2}^{var,5}$ of the set $M_S^{var,5}$).

2.15. If, for the cases of substeps 2.11, 2.12, when establishing communication between the system components, the standard part (according to the "flowering" scheme only to confirm the establishment of communication and the activity of the components) is completed successfully, and the additional part is not completed successfully. Then, the system components that have established such a fact about a certain component, such a result is reported to the decision-making center of the system (an element $m_{S,2}^{var,5}$ of the set $M_S^{var,5}$).

2.15.1. If such a message is received from two components that were trying to establish a connection with each other, then the decision center instructs them to retry the establishment of the connection in an additional part of the procedure, additionally instructing another component to establish a connection with these two components and these three components must inform the decision-making center about the performance results.

2.15.2. If such a message was received from one of the two components that attempted to establish communication with each other, then the decision center instructs this component and two other active components to attempt to establish communication according to an additional part of the procedure and the results of the execution of these three components must be provided to the decision-making center.

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

2.16. If it is confirmed in substeps 2.13.1 and 2.13.2 that there are problems with establishing communication with a certain component by an additional part of the communication establishment procedure, then such a component is investigated by the decision-making center (an element $m_{S,2}^{var,5}$ of the set $M_S^{var,5}$) through immediate testing of communication with it by a certain number of components. When problems are detected, it is removed from the system and a corresponding message about such an event is sent to the system administrator.

2.17. If the computer stations in which the system components are present are turned off correctly at the same time (an element $m_{S,1}^{var,6}$ of the set $M_S^{var,6}$), then the system components in them store information about the correct completion of their operation and start work with standard specified actions the next time.

2.18. If the computer stations, in which the system components are present, are disabled at the same time (an element $m_{S,2}^{var,6}$ of the set $M_S^{var,6}$), then the components did not complete the correct exit and when the computer systems are turned on, the components installed in them will perform the correct restart procedure execution of unfinished previous tasks along with the initial boot procedure.

2.19. If the computer stations, in which the system components are present, are turned off correctly at different times (an element $m_{S,3}^{var,6}$ of the set $M_S^{var,6}$), then the system components in them store information about the correct completion of their operation and the next time start work with standard specified actions, considering the time of turning off the rest of the components in relation to a certain component.

2.20. If the computer stations, which have system components, are turned off at different times partially correctly (element $m_{S,2}^{var,8}$ of the set $M_S^{var,8}$) and partially accidentally (element $m_{S,4}^{var,6}$ of the set $M_S^{var,6}$, element $m_{S,3}^{var,8}$ of the set $M_S^{var,8}$), then for the components that were in the computer stations that were turned off correctly, we perform substep 2.19 and substep 2.18.

Step 3. Ensuring system integrity.

3.1. If system S is divided into two or more unrelated subsystems within the corporate network because of equipment failure for a certain time, then each of the parts will reform itself into a reduced system S and will continue to work, provided that in each of the parts there are no less than two active components with a decision center.

3.1.1. If one of the parts does not have active components with a decision center and is inactive, then the components of this part block the operation of the computer stations and issue a corresponding message to the administrator.

3.1.2. If components with a decision center are inactive at the moment of a certain emergency or intentional separation of the second part, which will contain all active components with a decision center of the system, then their transfer to the active state will occur after another communication session and establish the absence of active components with the center decision-making.

3.2. If part of the active components, which contain the decision-making center of the system, is removed from the system for certain reasons, then the remaining part will start the procedure of forming the system from the existing components.

3.3. If there are no available active components with a decision center, the available components block the computer stations and issue a corresponding message to the administrator.

Step 4. Organization of partial centralization.

4.1. The formation of a decision regarding the number of components (element $m_{S,1}^{st}$ of the set M_S^{st}),) in which the decision-making center of the system will function is determined by all components of the system, in which the functionality of the decision-making center is available, at the first start of the system. The number of active components of the decision-making center will be less than two-thirds and more than one. Each component at the beginning of the start of the system randomly generates a number from the interval from two to two-thirds of the number of components of the center, and all these components exchange such numbers among themselves and find the average arithmetic number among these numbers and discard the fractional part in it.

4.2. If at the next start of the system not all components with the decision-making center of the system will be active in the enabled computer stations, then the available components will decide on the number of active components (element $m_{S,1}^{st}$ of the set M_S^{st}) in which the decision-making center will be located. When turning on computer stations with components in which the decision-making center was active at the previous stage of operation, such components receive a message from the decision-making center about the transition to the passive state of their decision-making center functionality.

4.3. To select certain components of the decision center to be active, after performing substep 4.1, each component randomly generates numbers from a range of one to a number equal to the number of components with the decision center. After the formation of such sequences of numbers, results are exchanged between all components with a decision-making center. In all sequences, the numbers are sorted in non-descending order, and after sorting, the whole part is calculated

from the arithmetic mean value of the numbers with the same index.

4.4. To select certain components of the decision center to be active, after performing substep 4.2, each component randomly generates a number from one to a number equal to the number of active components with the decision center at the current time. After the formation of such sequences of numbers, the results are exchanged between all active components with the decision-making center. In all sequences, the numbers are sorted by non-decreasing order, and after sorting, a whole part of the arithmetic mean value of the numbers with the same index is calculated.

4.5. If there are two such active components at substep 4.4, then they will perform the functionality of the decision-making center, and when components with the functionality of the decision-making center appear in the system, they will perform substep 4.4.

4.6. If there are less than two such active components at substep 4.4, then the functionality of the decision-making center will be in one component. Substep 4.5 will be performed when components with the functionality of the decision-making center appear in the system.

4.7. If the enabled computer stations have components that do not have decision center components, then each of the components records events, and the system does not function normally.

4.8. Tasks for the system regarding its further steps or for certain components are formed separately in each of the active components of the decision-making center, and after a decision has been agreed between them, such a task is notified for execution.

For the substeps of step 4, there may be other algorithms for determining the number of components with a decision-making center and directly components of a decision-making center. For example, there may be weighted averages, harmonic averages, etc. In addition, the functionality can contain several algorithms, and at the current time, all components can use one of them.

Step 5. Migration of the decision-making center of the system.

5.1. If all the components of the decision-making center are active at the current moment of time, some of them form the decision-making center, and the rest are in a passive state, then periodically some of the active components will become passive and vice versa passive components will become active. The decision on the next review of the components involved in the formation of the decision-making center will be made by the currently active components.

5.2. If the security status in the computer station has decreased according to the system assessment and the component in it is an active component of the decision-making center, then the rest of the system components decide to transfer this component to a passive state and make the other component active.

5.3. If not all components of the decision-making center are active at the current moment due to their computer stations not being turned on, some of the active ones form the decision-making center, and the remaining components with the decision-making center are in a passive state, then the decision-making center will supplement the number of active components at the expense of passives.

Step 6. Evaluation of the state of the components and the system.

6.1. We calculate the general states of the components and computer stations $\alpha'_{3,S_{1,n}}$ according to equation (53, [1]).

6.2. We calculate the state of the system as a whole according to equation (6).

Step 7. Evaluation of the results of the distributed calculations in components.

7.1. The values of the characteristic indicators of the system components, depending on the types of performed tasks, are determined by formulas (12, [1]), (44, [1]) and (53, [1]) and sent to all active components of the decision-making center.

7.2. If the results of calculations carried out in different components of the system are the same and each of the components participating in their processing received the same values during the specified time interval, then one of the obtained results is accepted as the final value of the distributed calculations.

7.3. If the results of the calculations performed in different components of the system are not the same and each of the components participating in their processing received the same set of values during the specified time interval, then the percentage of the largest number of identical values of the calculation results to all the received values is determined.

7.3.1. If the percentage of values of calculation results is equal to $\frac{N-1}{N} \cdot 100\%$ (N – number of values), then the component in which the result is different from the rest will be sent additional verification values for calculations to check its legitimacy, and one of the $N-1$ obtained results is accepted as the final value of distributed calculations. If the component that will be checked because of a different value of the result from the rest belongs to the active components of the decision center, then it will be sent for processing the received set of values, and the rest of the active components of the decision center will examine its response and make decisions about its further functioning in the system.

7.3.2. If the percentage of the values of the calculation results is less than $\frac{N-1}{N} \cdot 100\%$ (N – the number of values) and more than 50%, then the components in which the result is different from the rest will be sent

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

additional verification values for calculations to check its legitimacy, and one of the results obtained, which was more than 50, is accepted as the final value of distributed computing. If the component that will be checked because of a different value of the result from the rest belongs to the active components of the decision center, then it will be sent for processing the received set of values, and the rest of the active components of the decision center will examine its response and make decisions about its further functioning in the system.

7.3.3. If the percentage of values of the calculation results is less than 50% from the number of all among the largest number of one value, then the calculation results are not accepted and the system begins to perform self-testing. After completion, if successful, it will retry this task or reject its execution.

The value of the characteristic indicators of the system components, depending on the types of tasks performed can be determined by formulas (7) – (12) depending on the features of processing. Clusters of such values can be formed by taking into account time delays when transmitting the results of distributed calculations. Niche formulas for determining the values of the characteristic indicators of the system components may also be applicable.

Step 8. Determination of components in which the task set by the system will be performed.

8.1. To determine the components in which the task set by the system will be performed, we determine the security level of all components according to formula (5), according to the first variant of division into classes (formulas (7) – (10)) and, thus, determine half of the components, putting in the formula (10) factor 0.5 instead of 0.2.

8.2. If the assigned task requires immediate execution or has a status related to security research in components, it is performed by all components.

8.3. We involve a smaller number of components, if there are many of them, to perform the assigned task as required (by a decision made or by an instruction to perform), but this number cannot be less than ten components.

8.4. If the number of components in the system is small, e.g., less than ten, all components are involved in the performance of the assigned task.

8.5. Processing of accumulated information in the components of the decision-making center of the system, formation of the base of decisions made in these components and provision of such a base to all components of the decision-making center.

8.6. Decision to perform a specific function in components depending on the current data in the system.

Step 9. Reconstruction of the system architecture in the presence of critical events.

9.1. If anomalous events or malicious manifestations are detected in the computer network or stations and the components report them, the system architecture rebuilding procedure is launched.

9.2. If in certain components of the system, long-term functioning of subsystems for detecting anomalous events or malicious manifestations is detected, and at the same time, such components inform the decision-making center about the need to continue the execution of the task, and the time limits for the execution of such tasks have already been passed, then the system is determined with the need to rebuild its architecture without considering these components, and in their presence, the functionality of the decision-making center is transferred from an active state to a passive one.

9.3. If the system is in a critical state according to the calculated value of the security level, then it removes a part of the components with the largest values of the critical state from its architecture and recalculates the current state.

9.3.1. If after such a reconstruction, the security status is not critical, it continues to function.

9.3.2. If the security status remains critical after such a rebuild, it stops functioning and issues a message to the administrator.

Step 10. Determination of further steps of the system at the current time.

10.1. We determine the next state of system S through its current state and indicators of components and the system according to equation (53, [1]).

10.2 If an event from a set M_S^{pd} has occurred in the system, it is processed by the functions of the system components, and the decision-making center selects a state variant from a set of states M_S^{st}, evaluates the possibility of performing a specified transition to the next state at the current moment of time, and directly performs the transition with verification of its complete completion.

10.3. If an event occurred in the system that is not from the set M_S^{st}, it is processed by the functions of the decision-making center.

10.3.1. If a state variant is selected from a set of states, an evaluation of the possibility of performing a specified transition to the next state at the current moment of time is performed. The transition is then performed directly with a check of its complete completion, and the set of events is supplemented by this event.

10.3.2. If a state option is selected from a set of states M_S^{st}, an evaluation of the possibility of performing a specified transition to the next state at the current moment of time is carried out, then the transition is then carried out directly with a check of its complete completion. If the event remains active after the system state

changes, the system blocks the components and reports a problem administrator.

10.4. If among the elements of the set of events M_S^{pd} there is no event that occurred specifically in the system and needs to be processed, then the decision-making center of the system returns the system to its previous state and analyzes the presence of this event.

10.4.1. If the event is present after a change in the system state, then the system components block the processes in the computer stations, and the entire system transitions to a critical security state and add the event to the set of events.

10.4.2. If the event is absent after changing the state of the system to the previous one, then the system adds this event to the set of events and fixes the state of the system in which it disappears.

10.5. If the event, which is among the elements of the set of events M_S^{pd}, does not require transition to the next state, then it is processed and the system remains in the current state.

10.5.1 If part of the system components did not have time to complete the transition to the specified state for certain reasons, then in the future, when they are active, they reproduce the missed states in their history of states, renew the current indicators, and return to the current state of the system.

Step 11. Completing the functioning of the components and the system.

11.1. Completion of the functioning of the components and the system at the current moment in time with their subsequent return to the performance of tasks when the computer stations are turned on.

11.2. Blocking of part of the components or system by the decision-making center of the system.

11.3. Completion of the functioning of a part of the components at the current moment of time with their subsequent return to the performance of tasks when the computer stations are turned on.

11.4. Removal of components or the entire system from nodes in the network.

Thus, the developed method for organizing the functioning of partially centralized distributed systems makes it possible to create them according to the principles of self-organization and adaptability. Partial centralization of such distributed systems is achieved by separating the components of the decision-making center of the system, in each of which a decision is made separately, which is later coordinated with the rest of the decisions.

At the same time, the components of the decision-making center function according to the principle of decentralization, and the entire system functions according to the principle of centralization. In the developed method of functioning of this type of system, the distribution of components was carried out in relation to the decision-making center, which made it possible to implement partial centralization compatible with the principles of self-organization and adaptability.

5. Experiments

5.1. Experimental settings

The degree of degradation of system S in the process of its functioning and the degradation of its components will be considered in the context of the loss of some components by the system and, as a result, either the removal of some components irrevocably from the system or a decrease in the system's performance due to the loss of some components or their incorrect functioning.

The degree of system degradation correlates with the degree of sustainability [1, 40, 41]. However, the system's stability reflects the ability to continue functionality and fulfill its tasks despite changes in the operating environment with minimal change or loss of functionality, and degradation reflects the ability to fulfill its tasks after a complete or partial loss of component functionality and approaching or transitioning to both a state of failure and a state of complete shutdown. Thus, the common feature of both system characteristics is the ability to continue performing the assigned tasks. The difference is that stability is the probability of continuing operation, and degradation is the probability of approaching a state of failure.

The degree of degradation of the system as a whole will depend on the number of components in the system, the time of operation of the system and its components, the events that the system will process, and the impact on the environment in which the system and its components will operate. When determining the system degradation factor, we will consider the number of components in the system, the operating time of the system and components, and the values of the security levels of the components and the system at the current time. Then, we define the system degradation factor as follows:

$$k_{S,t}^{d} = 1 - \left(\frac{n}{k}\right)^{-\frac{\frac{\sum_{r=1}^{i}\alpha'_{1,S_i}}{k_1}+\frac{\sum_{r=i+1}^{n}\alpha'_{2,S_{k+1,n}}}{k_2}}{\alpha_{S,t}^{st,1}}\cdot t}, \qquad (14)$$

where $\alpha_{S,t}^{st,1}$ is the value of the system security level calculated by formula (12, [1]); k is the number of active components in the system; k_1 is the number of active components with a decision-making center in the system; k_2 is the number of active components without a decision-making center in the system; α'_{1,S_i}, $\alpha'_{2,S_{k+1,n}}$ – are the values of the security levels of the system components calcu-

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

lated by formulas [1]); $k = k_1 + k_2$; n is the number of components in the system; t is the system operation time.

If there are many components in the system, the loss of some of them will not significantly affect the degree of system degradation because the system architecture at the component level is centralized and, therefore, the tasks can be assigned to the remaining active components.

If the degradation indicator relates directly to a particular component, if it is at the stage of removal by the system itself, this does not significantly affect the degree of degradation of the entire system.

If all components of the system function normally, then $k = n$ and the degradation factor $k^d_{S,t} = 0$. If $k < n$, then the value of the degradation coefficient will be different from zero and will indicate the degree of system degradation.

Let's set up an experiment to determine the degree of system degradation.

The values of the significance levels of the characteristic indicators are the same as those in the previous experiment.

Consider system S at the level of its components. Two indicators can be received from each component to the system's decision-making center:

1) the value of the component's safety level;

2) the fulfillment of the task.

Then, four cases are possible:

1) the value of the component's safety level corresponds to the permissible value and the task is obtained correctly;

2) the safety level of the component corresponds to the permissible value, but the task is performed incorrectly;

3) the safety level of the component does not correspond to the permissible value and the task is performed incorrectly or not completed within the specified time;

4) the security level of the component does not correspond to the permissible value, but the task is performed correctly.

We will create a file of the results of work over a long period of time in the system, recording the results of the values of the components' security levels and the results of the tasks performed. After a certain period of system operation, we will process the results stored in the specified file.

5.2. Case study

The experiment was conducted in five series over five days.

In the first series of the experiment, out of 100 components of the system, 9 components produced a negative result, and the rest produced a positive result.

Thus, the task set during the experiment was correctly performed using 91 components.

In the second series of the experiment, out of 100 system components, 10 components produced a negative result and the rest produced a positive result.

In the third series of the experiment, out of 100 components of the system, 10 components gave a negative result and the rest gave a positive result.

In the fourth series of the experiment, out of 100 components of the system, 10 components gave a negative result and the rest gave a positive result.

In the fifth series of the experiment, out of 100 components of the system, 9 components produced a negative result, and the rest produced a positive result.

In addition, the security status of each of the 10 components was obtained from the given file for each series of experiments.

As a result, it was found that the negative result in the series of the experiment was obtained from components with a high and low level of security. This affected the system degradation rate. The fewer components with a high level of security and a negative result of the task, the lower the level of system degradation at the current time.

To assess the approach performance, ROC analysis was used [42, 43]. It provides a more comprehensive view of a method's performance than a single accuracy score, helping us to make informed decisions about method selection and parameter tuning [44, 45].

The results of the experiment are presented in the form of ROC curve graphs and tables of the values of security levels and task performance. The values of the degree of system degradation for the series of experiments are given in Tables 1–21 and Figs. 4–13.

Table 1

The value of the degree of system degradation

	Series of the experiment					Average value
	1	2	3	4	5	
Degradation factor	0.15	0.25	0.21	0.21	0.27	0.21

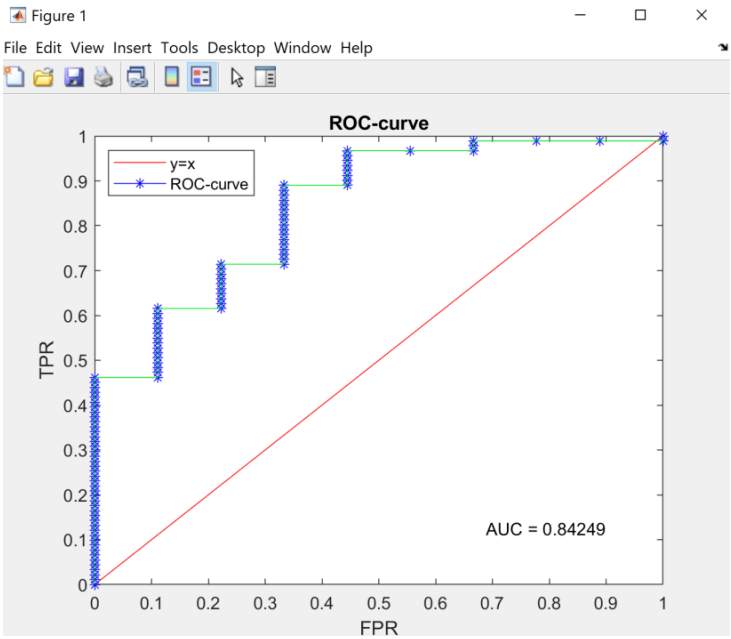

Fig.4. Experiment 1.1 (9 zeros)

Table 2

Results of experiment 1.1: 1– detected, 0 – not detected

Network host	Detection result									
1-10	1	1	1	1	1	1	0	1	1	1
11-20	1	1	1	1	1	1	1	1	1	1
21-30	1	1	1	1	1	0	1	0	1	1
31-40	0	1	1	1	1	1	1	1	0	1
41-50	0	1	1	1	1	1	1	1	1	1
51-60	1	1	1	0	1	1	1	1	1	0
61-70	1	1	1	1	1	1	1	1	1	1
71-80	1	1	1	1	1	1	1	1	1	1
81-90	1	1	1	1	1	1	1	1	1	1
91-100	1	1	1	1	1	0	1	1	1	1

Table 3

Results of experiment 1.1

Network host	Statistical probability of malware detection									
1-10	0.9860	0.9860	0.9930	0.9740	0.9888	0.9804	0.9800	0.8748	0.9959	0.9760
11-20	0.9900	0.9970	0.976	0.9790	0.9849	0.9967	0.9850	0.9783	0.9860	0.9872
21-30	0.9820	0.9700	0.919	0.9850	0.9814	0.5994	0.9808	0.9770	0.9761	0.9944
31-40	0.6730	0.9720	0.9810	0.2700	0.9051	0.9974	0.9796	0.9799	0.9084	0.9930
41-50	0.1340	0.9990	0.9750	0.9790	0.9906	0.9742	0.9854	0.9916	0.9979	0.9920
51-60	0.7880	0.9820	0.9770	0.9860	0.8780	0.9953	0.9899	0.9945	0.9938	0.1889
61-70	0.9793	0.4386	0.9996	0.9931	0.9029	0.9912	0.9879	0.9926	0.9849	0.8906
71-80	0.9720	0.9991	0.9730	0.9864	0.9821	0.9732	0.9917	0.9884	0.7225	0.9870
81-90	0.9943	0.9873	0.9983	0.0671	0.9533	0.9937	0.9842	0.9949	0.9797	0.9993
91-100	0.9783	0.9722	0.9554	0.9949	0.9977	0.2542	0.9941	0.9861	0.9839	0.9293

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

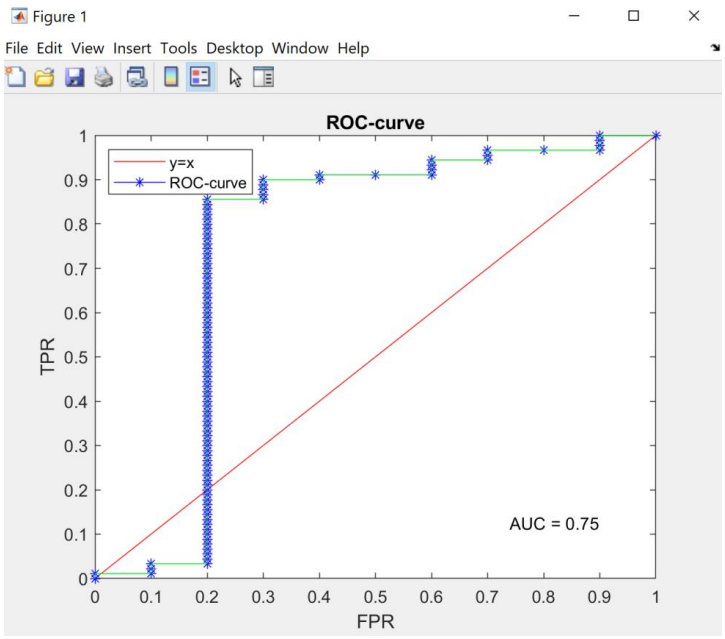

Fig.5. Experiment 1.2 (10 zeros)

Table 4

Results of experiment 1.2: 1– detected, 0 – not detected

Network host	Detection result									
1-10	1	1	1	1	1	1	1	1	1	1
11-20	1	1	1	1	1	1	1	1	0	1
21-30	0	1	0	1	1	0	1	1	1	1
31-40	1	1	1	1	1	1	1	1	1	1
41-50	1	1	1	1	1	1	1	0	1	1
51-60	1	1	1	0	1	1	1	1	1	1
61-70	1	1	1	1	1	1	1	1	1	1
71-80	1	1	0	1	0	1	1	1	1	1
81-90	1	1	1	1	1	1	1	1	1	1
91-100	1	1	1	1	0	1	0	1	1	1

Table 5

Results of experiment 1.2

Network host	Statistical probability of malware detection									
1-10	0.9844	0.9905	0.9762	0.9882	0.9641	0.9964	0.9740	0.9233	0.9988	0.9746
11-20	0.9746	0.9747	0.9727	0.9836	0.9901	0.9949	0.7714	0.2393	0.9228	0.9913
21-30	0.9987	0.9852	0.9108	0.9937	0.7070	0.7605	0.8793	0.9886	0.9885	0.9737
31-40	0.9737	0.9785	0.9921	0.9823	0.9949	0.9981	0.9820	0.9716	0.9871	0.9924
41-50	0.9796	0.9848	0.9766	0.9982	0.9845	0.9862	0.9766	0.0012	0.9718	0.9946
51-60	0.9931	0.9759	0.9969	0.9345	0.9897	0.9358	0.9710	0.9827	0.9847	0.9875
61-70	0.9725	0.9898	0.9401	0.9867	0.9914	0.9846	0.9885	0.9456	0.9894	0.9814
71-80	0.9731	0.2410	0.5323	0.9772	0.4249	0.9857	0.9824	0.9765	0.9958	0.9958
81-90	0.9785	0.9885	0.9934	0.9105	0.5369	0.9815	0.9749	0.9939	0.9734	0.9748
91-100	0.3235	0.9954	0.9875	0.9876	0.9978	0.9873	0.9703	0.9943	0.9883	0.9844

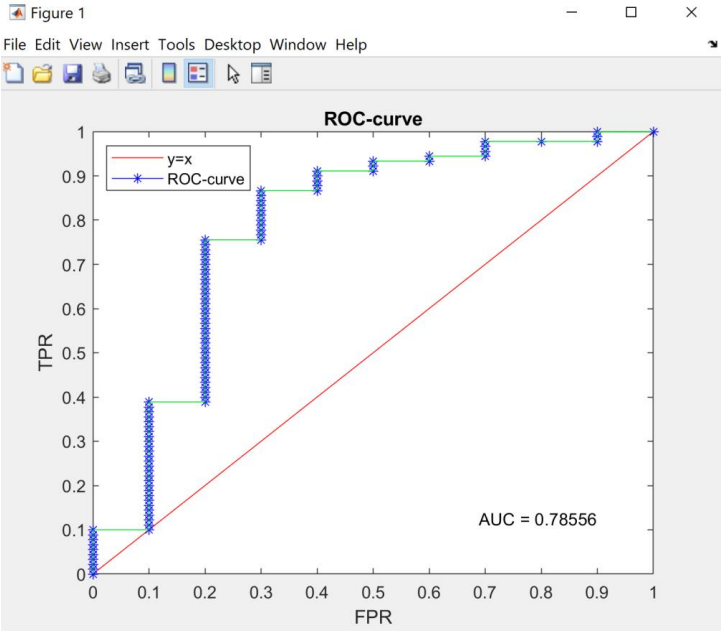

Fig.6. Experiment 1.3 (10 zeros)

Table 6

Results of experiment 1.3: 1– detected, 0 – not detected

Network host	Detection result									
1-10	1	1	1	1	1	1	1	1	1	1
11-20	1	1	1	1	1	1	1	1	1	1
21-30	1	1	1	0	1	1	1	1	1	1
31-40	1	1	1	1	1	0	1	1	1	1
41-50	1	0	1	1	1	1	1	1	0	1
51-60	1	1	1	1	1	1	1	1	0	1
61-70	0	1	1	1	1	1	1	1	1	1
71-80	1	1	1	1	1	1	1	1	1	1
81-90	1	1	1	0	1	1	0	1	1	1
91-100	1	1	0	1	1	1	1	1	0	1

Table 7

Results of experiment 1.3

Network host	Statistical probability of malware detection									
1-10	0.9885	0.9976	0.9881	0.9911	0.9923	0.9816	0.9775	0.9711	0.9842	0.9894
11-20	0.9784	0.9855	0.9774	0.9789	0.9248	0.9967	0.9958	0.9763	0.9820	0.7904
21-30	0.9777	0.9990	0.9886	0.3202	0.9948	0.9897	0.9864	0.9775	0.9712	0.9770
31-40	0.9808	0.9890	0.9996	0.9762	0.9927	0.9530	0.3404	0.9748	0.9943	0.9843
41-50	0.9735	0.9963	0.9891	0.1759	0.9973	0.9711	0.9712	0.9997	0.0510	0.3235
51-60	0.9851	0.9929	0.9175	0.9918	0.9910	0.9838	0.3394	0.9802	0.9751	0.9820
61-70	0.2833	0.9768	0.9808	0.0895	0.9725	0.9854	0.9950	0.9971	0.9917	0.9815
71-80	0.9789	0.9908	0.9964	0.8739	0.9724	0.9845	0.9738	0.9776	0.9965	0.9142
81-90	0.9352	0.9863	0.8711	0.4908	0.9937	0.9952	0.9116	0.9825	0.9893	0.9764
91-100	0.9885	0.9903	0.9880	0.9804	0.9809	0.9751	0.9939	0.9654	0.8532	0.9933

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

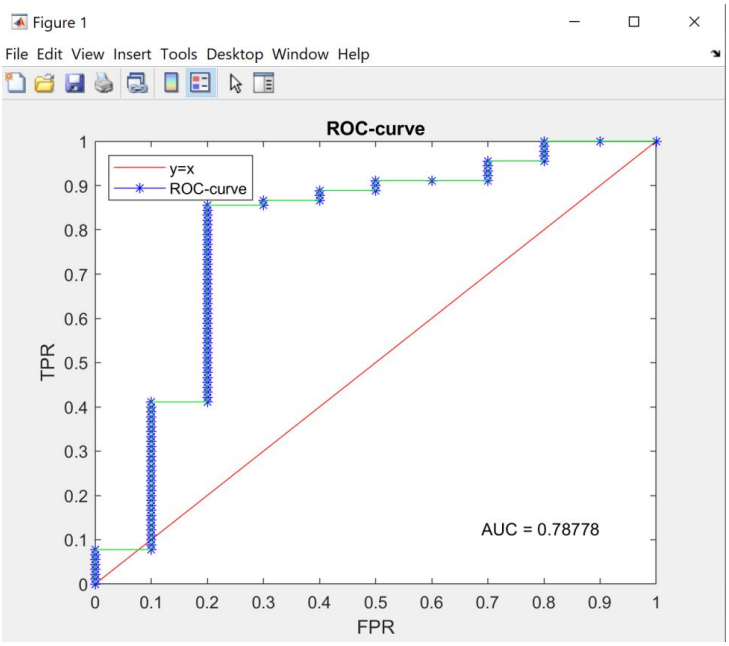

Fig.7. Experiment 1.4 (10 zeros)

Table 8

Results of experiment 1.4: 1– detected, 0 – not detected

Network host	Detection result									
1-10	1	1	1	1	1	1	1	1	1	1
11-20	1	1	0	1	0	1	1	1	0	1
21-30	1	1	1	1	1	0	1	1	1	1
31-40	1	1	0	1	1	1	1	1	1	1
41-50	1	1	1	1	1	1	1	1	1	1
51-60	1	1	1	1	1	1	1	0	1	1
61-70	0	1	1	1	1	1	1	1	1	1
71-80	1	1	1	1	1	1	1	1	1	1
81-90	1	0	1	1	1	1	1	0	1	0
91-100	1	1	1	1	1	1	1	1	1	1

Table 9

Results of experiment 1.4

Network host	Statistical probability of malware detection									
1-10	0.9716	0.9850	0.9830	0.9971	0.9889	0.9995	0.9876	0.9952	0.9841	0.9864
11-20	0.9754	0.9890	0.8815	0.9860	0.9844	0.9494	0.9728	0.9964	0.9701	0.4440
21-30	0.9904	0.9870	0.9844	0.9796	0.9880	0.0566	0.9905	0.9984	0.9730	0.9853
31-40	0.9733	0.9864	0.5237	0.3030	0.9933	0.9820	0.9969	0.9569	0.9718	0.2197
41-50	0.9725	0.9985	0.9705	0.9734	0.9704	0.9765	0.8759	0.9893	0.9855	0.9774
51-60	0.9758	0.9727	0.9811	0.6721	0.9881	0.9844	0.3351	0.9024	0.9952	0.9779
61-70	0.1017	0.9834	0.9802	0.9952	0.9995	0.9089	0.9754	0.9737	0.9874	0.9799
71-80	0.9780	0.9865	0.9754	0.9904	0.9717	0.9710	0.6912	0.9624	0.9970	0.9954
81-90	0.9819	0.9421	0.9829	0.9825	0.9919	0.9822	0.9986	0.9974	0.9985	0.9576
91-100	0.9787	0.9966	0.9763	0.9739	0.6408	0.9972	0.9821	0.9160	0.9724	0.9980

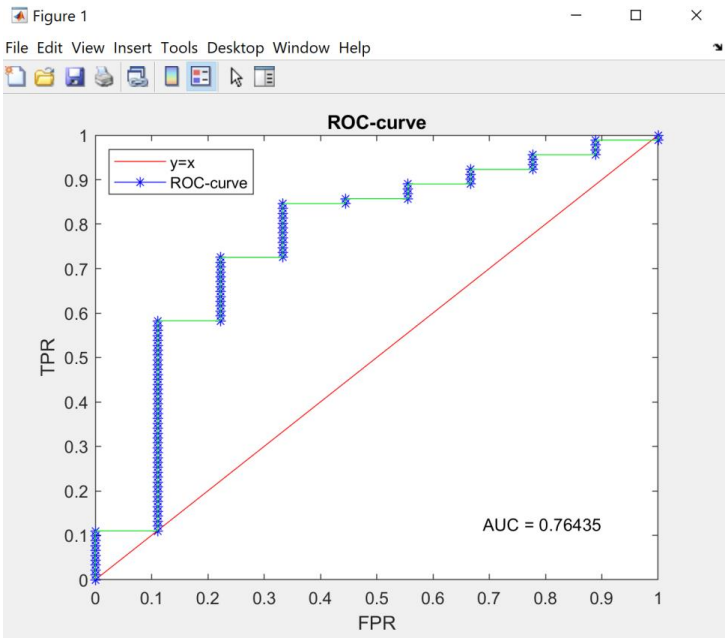

Fig.8. Experiment 1.5 (9 zeros)

Table 10

Results of experiment 1.5: 1– detected, 0 – not detected

Network host	Detection result									
1-10	1	1	0	1	1	1	1	1	1	1
11-20	1	1	0	1	1	1	0	1	1	0
21-30	1	1	1	1	1	1	1	1	1	1
31-40	1	1	1	1	1	1	1	1	1	0
41-50	1	1	1	0	1	1	1	1	1	0
51-60	1	1	1	1	1	1	1	1	1	1
61-70	1	1	1	1	1	1	0	1	1	1
71-80	1	1	1	1	1	1	1	1	1	1
81-90	1	1	1	1	1	1	1	1	0	1
91-100	1	1	1	1	1	1	1	1	1	1

Table 11

Results of experiment 1.5

Network host	Statistical probability of malware detection									
1-10	0.0061	0.9888	0.8821	0.9968	0.9764	0.9701	0.9964	0.9771	0.9773	0.9892
11-20	0.9791	0.8605	0.9965	0.4362	0.9817	0.9940	0.9494	0.9888	0.9910	0.4073
21-30	0.9859	0.9967	0.9779	0.9770	0.9952	0.9849	0.9746	0.9769	0.9095	0.9869
31-40	0.9788	0.9887	0.9915	0.9784	0.9824	0.9809	0.9934	0.9741	0.9971	0.9787
41-50	0.9850	0.9935	0.9903	0.9745	0.5002	0.9739	0.9984	0.9966	0.9855	0.0363
51-60	0.9993	0.2872	0.9395	0.9948	0.9934	0.9757	0.9829	0.9704	0.9798	0.9740
61-70	0.9835	0.9872	0.9938	0.9826	0.9860	0.8861	0.9379	0.9863	0.8905	0.9716
71-80	0.9943	0.9800	0.9769	0.9947	0.9063	0.9750	0.9708	0.9987	0.9904	0.9958
81-90	0.9201	0.1974	0.8860	0.9777	0.9966	0.9976	0.9790	0.9722	0.8993	0.9725
91-100	0.9919	0.9834	0.3147	0.9751	0.9859	0.9890	0.9704	0.9841	0.9966	0.9734

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

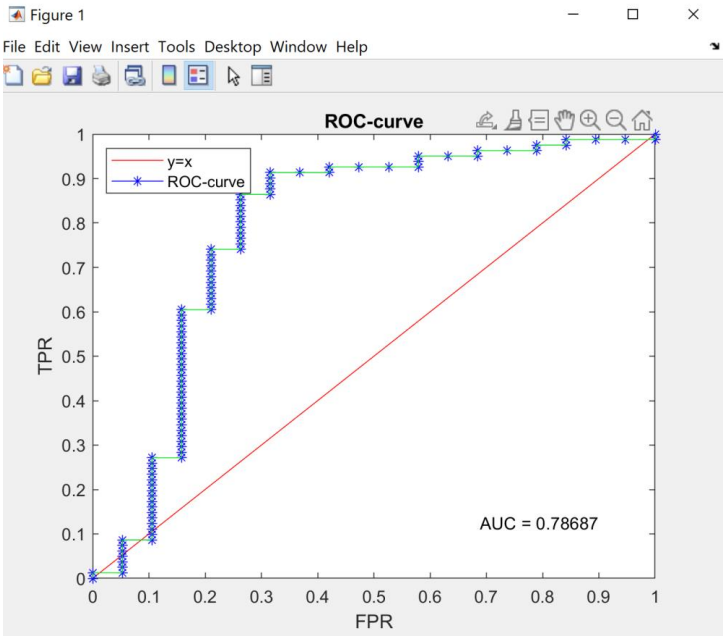

Fig.9. Experiment 2.1 (19 zeros)

Table 12

Results of experiment 2.1: 1– detected, 0 – not detected

Network host	Detection result									
1-10	1	1	1	1	1	1	0	1	1	1
11-20	1	1	1	1	1	1	1	1	1	1
21-30	1	0	1	1	1	0	0	0	0	0
31-40	1	0	1	1	1	1	1	1	1	1
41-50	1	0	1	1	1	1	1	1	1	1
51-60	1	1	1	1	1	1	1	1	1	1
61-70	1	1	0	1	0	1	1	1	0	1
71-80	1	1	1	1	1	1	1	1	1	0
81-90	1	1	0	1	0	1	1	0	1	1
91-100	1	1	1	1	1	1	1	0	0	0

Table 13

Results of experiment 2.1

Network host	Statistical probability of malware detection									
1-10	0.9832	0.9864	0.5353	0.9819	0.9925	0.9857	0.6537	0.9727	0.9775	0.9834
11-20	0.9891	0.9913	0.9998	0.9980	0.9728	0.8728	0.9749	0.9991	0.9879	0.9772
21-30	0.9721	0.9790	0.9944	0.9723	0.9806	0.4872	0.9202	0.1254	0.9911	0.8648
31-40	0.6905	0.9752	0.9895	0.9850	0.9785	0.9949	0.9946	0.9981	0.9700	0.9892
41-50	0.9702	0.8504	0.9732	0.9810	0.9772	0.9804	0.9775	0.9816	0.9826	0.9892
51-60	0.9936	0.9781	0.9953	0.9922	0.9948	0.9755	0.8839	0.9883	0.9910	0.9733
61-70	0.9729	0.9879	0.9281	0.9944	0.3046	0.9276	0.9971	0.9836	0.9721	0.9772
71-80	0.9920	0.9712	0.9827	0.9862	0.9986	0.0465	0.9735	0.9894	0.9733	0.9995
81-90	0.9775	0.9882	0.9266	0.9949	0.9492	0.9928	0.9975	0.9970	0.9764	0.9864
91-100	0.9935	0.9758	0.9924	0.9843	0.9875	0.4382	0.9725	0.5992	0.2718	0.9210

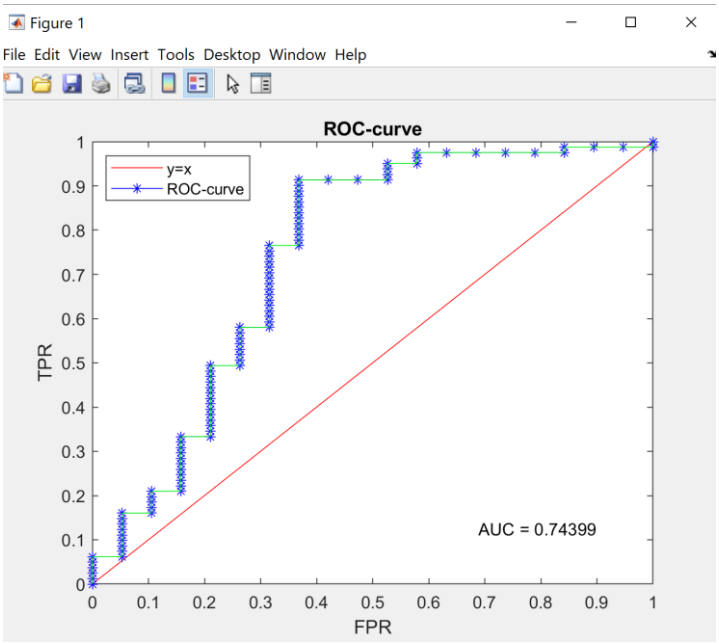

Fig.10. Experiment 2.2. (19 zeros)

Table 14

Results of experiment 2.2: 1– detected, 0 – not detected

Network host	Detection result									
1-10	1	0	1	1	1	1	1	1	1	1
11-20	1	0	1	0	1	1	1	1	1	1
21-30	1	1	1	1	1	1	1	1	1	1
31-40	0	1	1	1	0	1	0	1	0	1
41-50	1	1	1	1	1	0	1	1	1	1
51-60	0	1	1	0	0	1	1	1	1	1
61-70	1	0	1	1	1	1	1	1	1	1
71-80	0	0	1	1	1	1	1	1	1	1
81-90	0	1	1	0	1	1	1	1	1	1
91-100	1	1	1	0	1	1	1	0	0	1

Table 15

Results of experiment 2.2

Network host	Statistical probability of malware detection									
1-10	0.9751	0.9899	0.9861	0.9949	0.9780	0.9753	0.8889	0.9843	0.9936	0.9739
11-20	0.9715	0.0874	0.9709	0.9741	0.9908	0.9855	0.9863	0.9943	0.8618	0.9851
21-30	0.9783	0.9736	0.9966	0.9991	0.9983	0.9891	0.9727	0.9722	0.9032	0.9710
31-40	0.4778	0.9743	0.9891	0.9937	0.9166	0.9813	0.9946	0.9791	0.9796	0.4640
41-50	0.9851	0.9778	0.9920	0.9749	0.9976	0.2077	0.9725	0.9722	0.9931	0.9945
51-60	0.5608	0.9927	0.9988	0.9840	0.9936	0.9827	0.9983	0.9700	0.9994	0.9871
61-70	0.9804	0.8812	0.9790	0.9748	0.9900	0.9905	0.9938	0.9805	0.9775	0.9804
71-80	0.4204	0.9978	0.9927	0.9786	0.9882	0.9930	0.9954	0.9971	0.9879	0.0053
81-90	0.7280	0.9961	0.9824	0.9102	0.9935	0.9774	0.9866	0.9769	0.9452	0.9003
91-100	0.8660	0.9818	0.9776	0.9108	0.9899	0.9974	0.9702	0.5273	0.5088	0.9972

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

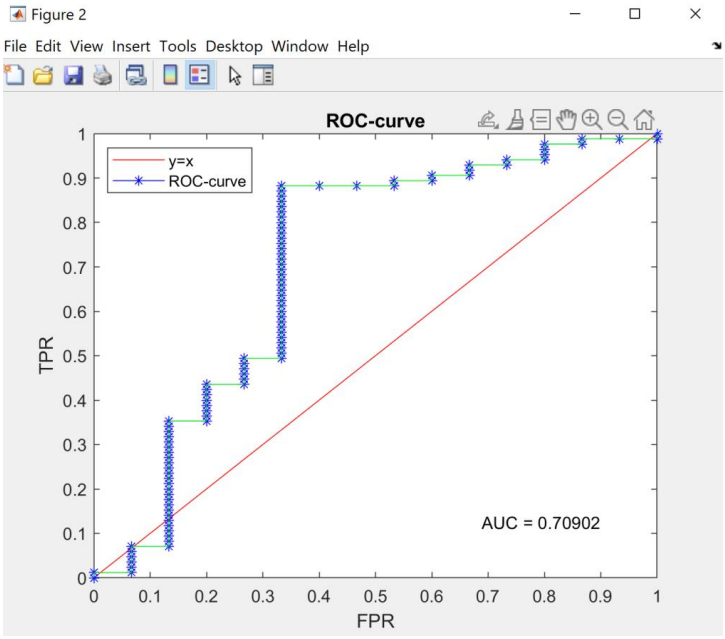

Fig.11. Experiment 2.3 (15 zeros)

Table 16

Results of experiment 2.3: 1– detected, 0 – not detected

Network host	Detection result									
1-10	1	1	1	1	1	1	0	1	1	1
11-20	0	1	1	0	1	1	0	1	1	1
21-30	1	1	1	1	0	1	1	1	0	0
31-40	0	1	0	1	1	1	1	1	1	1
41-50	1	1	1	1	1	1	1	1	1	1
51-60	1	1	0	1	1	1	1	1	1	1
61-70	1	1	1	1	1	1	1	1	1	1
71-80	1	1	1	1	1	1	0	1	0	1
81-90	1	1	1	1	1	0	0	1	1	1
91-100	0	1	1	1	1	1	1	1	1	1

Table 17

Results of experiment 2.3

Network host	Statistical probability of malware detection									
1-10	0.9970	0.9741	0.9938	0.9757	0.9709	0.9738	0.3687	0.9738	0.9981	0.9782
11-20	0.9983	0.9891	0.9962	0.9810	0.5633	0.9756	0.8612	0.9777	0.9792	0.9705
21-30	0.9876	0.9989	0.9955	0.9702	0.5038	0.9808	0.9734	0.9862	0.9825	0.9419
31-40	0.9966	0.9745	0.8740	0.9718	0.9814	0.9917	0.9729	0.9900	0.9789	0.9880
41-50	0.9746	0.9831	0.9704	0.8985	0.9779	0.9853	0.9765	0.9804	0.9924	0.9824
51-60	0.9717	0.9817	0.9400	0.9948	0.9791	0.9947	0.1103	0.9716	0.8662	0.9877
61-70	0.9844	0.9760	0.9772	0.9934	0.9885	0.5457	0.9915	0.9820	0.9839	0.9912
71-80	0.9820	0.9704	0.3890	0.9877	0.9834	0.9978	0.2787	0.5436	0.9864	0.9734
81-90	0.9971	0.9253	0.9972	0.9889	0.9704	0.6856	0.9599	0.9889	0.9718	0.9902
91-100	0.9085	0.8730	0.9855	0.9912	0.9944	0.7198	0.9793	0.9803	0.9900	0.9958

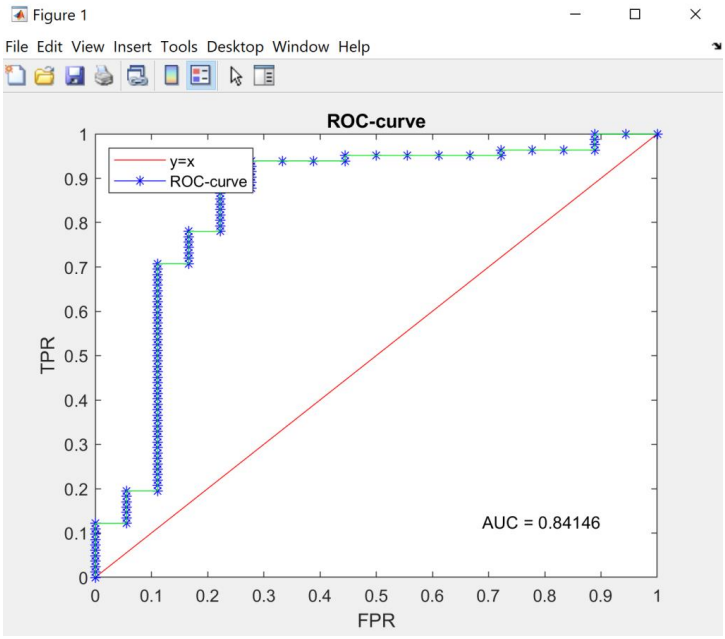

Fig.12. Experiment 2.4 (18 zeros)

Table 18

Results of experiment 2.4: 1– detected, 0 – not detected

Network host	Detection result									
1-10	1	1	1	1	1	1	1	1	1	0
11-20	1	1	1	1	0	1	1	1	1	0
21-30	1	1	0	0	1	0	1	1	1	1
31-40	1	1	1	1	1	1	1	0	1	1
41-50	1	1	1	1	1	1	1	0	1	1
51-60	1	1	1	1	1	1	1	0	1	0
61-70	1	1	1	1	1	1	1	0	0	1
71-80	1	1	1	1	1	0	1	1	1	1
81-90	1	0	1	1	1	0	1	1	1	1
91-100	1	1	1	0	0	1	1	0	1	1

Table 19

Results of experiment 2.4

Network host	Statistical probability of malware detection									
1-10	0.9765	0.9912	0.9712	0.9885	0.9901	0.9711	0.9701	0.8980	0.9959	0.9783
11-20	0.9860	0.9857	0.9870	0.9800	0.7357	0.9368	0.9995	0.9717	0.9819	0.9937
21-30	0.9878	0.9793	0.7196	0.9211	0.9796	0.9966	0.9897	0.2062	0.9842	0.3416
31-40	0.9985	0.9965	0.9831	0.7355	0.9798	0.9810	0.9938	0.9022	0.9986	0.9700
41-50	0.9789	0.9715	0.9833	0.9937	0.9974	0.9860	0.9941	0.8740	0.9925	0.9703
51-60	0.9843	0.9775	0.9792	0.9990	0.9763	0.9512	0.9768	0.4622	0.9999	0.9740
61-70	0.9986	0.9737	0.9756	0.9894	0.9738	0.9724	0.9898	0.9708	0.4407	0.9862
71-80	0.9812	0.9912	0.9984	0.9815	0.9908	0.1746	0.9933	0.9878	0.9813	0.9955
81-90	0.9768	0.8746	0.9999	0.9784	0.9913	0.9169	0.9824	0.9849	0.2310	0.9986
91-100	0.9920	0.9815	0.9712	0.0720	0.8537	0.9807	0.9964	0.8598	0.9887	0.9789

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

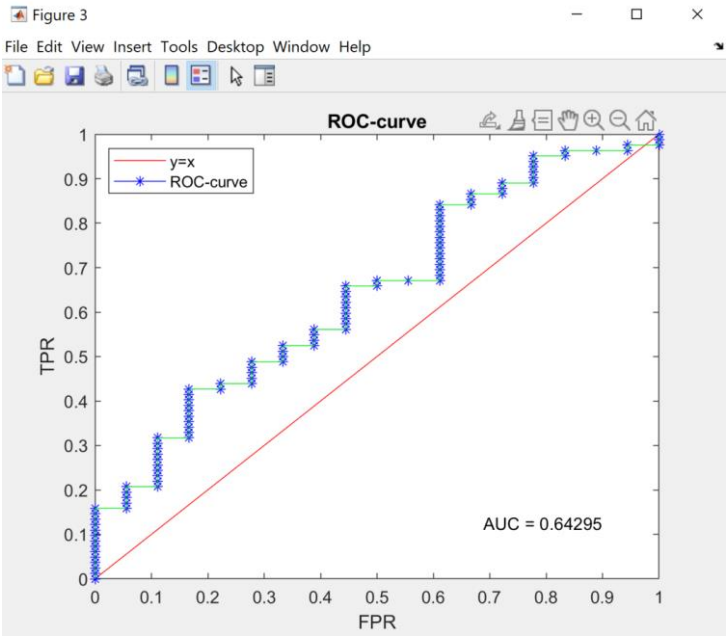

Fig.13. Experiment 2.5 (18 zeros)

Table 20

Results of experiment 2.5: 1– detected, 0 – not detected

Network host	Detection result									
1-10	1	1	1	1	1	1	1	1	1	1
11-20	1	1	1	1	1	1	1	1	0	1
21-30	0	1	1	1	1	1	1	1	1	1
31-40	0	1	1	1	1	1	0	1	0	1
41-50	1	1	1	0	0	1	1	1	1	1
51-60	1	0	1	1	0	1	1	0	1	1
61-70	1	1	1	1	1	1	1	1	1	1
71-80	1	0	1	1	0	0	1	1	1	0
81-90	0	1	1	1	1	1	1	1	1	1
91-100	1	1	0	0	0	1	1	1	1	1

Table 21

Results of experiment 2.5

Network host	Statistical probability of malware detection									
1-10	0.9939	0.8900	0.9704	0.9728	0.9126	0.9792	0.2426	0.9749	0.9833	0.9930
11-20	0.9501	0.9914	0.9838	0.9976	0.8607	0.9980	0.9838	0.9971	0.9816	0.9881
21-30	0.3499	0.9954	0.9785	0.6896	0.9881	0.9897	0.4901	0.9799	0.1005	0.9730
31-40	0.9786	0.9806	0.9861	0.9997	0.9708	0.9913	0.1580	0.9960	0.9736	0.9987
41-50	0.9832	0.9963	0.9959	0.9807	0.9160	0.9959	0.9706	0.9723	0.9813	0.9745
51-60	0.9710	0.5165	0.8680	0.9945	0.8908	0.9903	0.9963	0.9927	0.9769	0.9808
61-70	0.9809	0.1474	0.9801	0.9726	0.9835	0.9836	0.9709	0.9891	0.9718	0.9751
71-80	0.5900	0.9637	0.9702	0.9786	0.9813	0.9744	0.9722	0.8968	0.9811	0.9947
81-90	0.9861	0.9945	0.9838	0.9704	0.9701	0.9750	0.9810	0.9915	0.9748	0.9784
91-100	0.9893	0.9786	0.9797	0.9747	0.4821	0.9969	0.9966	0.9818	0.9490	0.9776

5.3. Discussion

For the second task, a similar series of experiments were conducted. The number of negative results was 19, 19, 15, 18, 18, respectively, for five series of the experiments. The degree of system degradation did not increase. This indicates that the system architecture provides a low level of degradation.

The results of the experiment confirm that the system can perform the task regardless of the components with a high degree of security, which gave a negative result, and the components with a low level of security, which also gave a negative result.

Most of the system components performed the task correctly and evaluated the results of all components to provide a consistent solution.

Conclusion and Future Work

The developed principle of the synthesis of multicomputer systems with combined baits and traps and a decision-making controller for detecting and counteracting malware and computer attacks is the basis of the concept of creating such systems. To detail the architecture of multicomputer systems with combined baits and traps and a decision-making controller for detecting and counteracting malware and computer attacks, which corresponds to the proposed principle of synthesis of such systems, it is necessary to develop a conceptual model of its architecture. The implementation of a decision-making controller through the development of a method for synthesizing systems with a controller will be the direction of further research.

Ensuring the organization of the functioning of partially centralized distributed systems, as one of the types of systems defined according to the developed principle, in computer networks is implemented by two developed methods.

The developed method of synthesis of mathematical models of security levels of system components makes it possible to obtain new mathematical models [1] of security levels of system components for a comprehensive description of processes that will take place in partially distributed systems and will be related to the evaluation of security of system components. It can be applied to discrete and continuous characteristic indicators. According to them, the values of the characteristic indicators of the security levels in the system components will be used to evaluate the results of distributed calculations obtained from various system components to determine the degree of trust in them.

The method for organizing the functioning of partially centralized distributed systems makes it possible to create such systems. For the operation of this type of system, the distribution of components was carried out according to the relationship to the decision-making center, which made it possible to implement partial centralization compatible with the principles of self-organization and adaptability, which set mechanisms for independent decision-making regarding further steps in the system and restructuring of its architecture as needed.

Thus, partially centralized distributed systems can be created using the two developed methods and filled with specialized functionality.

The direction of further research will be the development of specialized methods and their implementation in partially centralized systems, which can be deception systems, network baits, and narrowly specialized systems for detecting malicious software.

Contribution of the authors

Antonina Kashtalian is the developer of the principle of synthesis of multicomputer systems with combined baits and traps and a decision-making controller for detecting and counteracting malware and computer attacks. She also analyzed known methods of developing systems with deception technologies and their elements. and performed experimental studies with the system. The concept of deception systems and the principles of their operation, particularly distributed systems with varying degrees of centralization and conducted experimental studies with the system. Systems that are developed according to the method for organizing the functioning of partially centralized distributed systems are partial implementations of full-featured deception systems.

Sergii Lysenko described the research problem, determined the limitations of the subject area, and conducted an analysis of the relevance of the development of the scientific problem of the synthesis of such systems.

Bohdan Savenko is the developer of the method of organizing the functioning of partially centralized distributed systems for detecting malware. He has written the introduction section and conclusions of the paper, developed analytical expressions, and processed the results of experiments.

Tomas Sochor and **Tetiana Kysil** checked the analytical dependencies and set up the experiment.

All the authors have read and agreed to the published version of this manuscript.

References

1. Lysenko, S., & Savenko, B. Distributed Discrete Malware Detection Systems Based on Partial Centralization and Self-Organization. *International Journal*

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

of Computing, 2023, vol. 22, no, 2. pp. 117-39. DOI: 10.47839/ijc.22.2.3082.

2. Breeden, J. *5 top deception tools and how they ensnare attackers*. Available at: https://www.csoonline.com/article/570063/5-top-deception-tools-and-how-they-ensnare-attackers.html (accessed 06.08.2023).

3. *Acalvio ShadowPlex. Autonomous Deception*. Available at: https://www.acalvio.com/product/ 04.09.2023 (accessed 06.08.2023).

4. *SentinelOne*. Available at: https://www.sentinelone.com/surfaces/identity/ (accessed 06.08.2023).

5. *Proofpoint Identity Threat Defense*. Available at: https://www.proofpoint.com/us/illusive-is-now-proofpoint (accessed 06.08.2023).

6. *Counter Craft Security*. Available at: https://www.countercraftsec.com/ (accessed 06.08.2023).

7. *Fidelis Security*. Available at: https://fidelissecurity.com/fidelis-elevate/ (accessed 06.08.2023).

8. *The Commvault Data Protection Platform*. Available at: https://www.commvault.com/ (accessed 06.08.2023).

9. *Labyrinth Deception Platform*. Available at: https://labyrinth.tech/platform (accessed 06.08.2023).

10. *Labyrinth Deception Platform. Datasheet*. Available at: https://labyrinth.tech/assets/media/ pdf/labyrinth-data-sheet.pdf (accessed 06.08.2023).

11. Feng, M., Xiao, B., Yu, B., Qian, J., Zhang, X., Chen, P., & Li, B. A Novel Deception Defense-Based Honeypot System for Power Grid Network. *International Conference on Smart Computing and Communication*, 2021, Vol. 13202, pp. 297-307. Cham: Springer International Publishing. DOI: 10.1007/978-3-030-97774-0_27.

12. Walter, E., Ferguson-Walter, K., & Ridley, A. Incorporating deception into cyberbattlesim for autonomous defense. 2021. *arXiv preprint arXiv:2108.13980*. DOI: 10.48550/arXiv.2108.13980.

13. Anwar, A. H., Kamhoua, C. A., Leslie, N. O., & Kiekintveld, C. Honeypot Allocation for Cyber Deception Under Uncertainty. *IEEE Transactions on Network and Service Management*, 2022, vol. 19. no. 3, pp. 3438-3452. DOI: 10.1109/TNSM.2022.3179965.

14. Sayed, M. A., Anwar, A. H., Kiekintveld, C., & Kamhoua, C. Honeypot Allocation for Cyber Deception in Dynamic Tactical Networks: A Game Theoretic Approach. *14th International Conference on Decision and Game Theory for Security*. *GameSec 2023*. 2023. arXiv preprint. arXiv:2308.11817. DOI: 10.48550/arXiv.2308.11817.

15. Anwar, A. H., & Kamhoua, C. A. Cyber Deception using Honeypot Allocation and Diversity: A Game Theoretic Approach. *2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC)*, Las Vegas, NV, USA, 2022, pp. 543-549. DOI: 10.1109/CCNC49033.2022.9700616.

16. Anwar, A. H., Kamhoua, C., & Leslie, N. Honeypot allocation over attack graphs in cyber deception games. *International Conference on Computing, Networking and Communications (ICNC)*, 2020, pp. 502-506, IEEE. DOI: 10.1109/ICNC47757.2020. 9049764.

17. Acosta, J. C., Basak, A., Kiekintveld, C., & Kamhoua, C. Lightweight On-Demand Honeypot Deployment for Cyber Deception. In Gladyshev, P., Goel, S., James, J., Markowsky, G., Johnson, D. (eds) Digital Forensics and Cyber Crime. ICDF2C 2021. *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, 2022, vol. 441, pp. 294-312. Springer, Cham. DOI: 10.1007/978-3-031-06365-7_18.

18. Priya, D., & Chakkaravarthy, S. Containerized cloud-based honeypot deception for tracking attackers. *Scientific Reports*, 2023, vol. 13. DOI: 10.1038/s41598-023-28613-0.

19. Al-Shaer, E., Wei, J., Hamlen, K. W., & Wang, C. Autonomous Cyber Deception. Reasoning. Adaptive Planning. and Evaluation of HoneyThings. Springer Nature Switzerland AG, 2019. DOI: 10.1007/978-3-030-02110-8.

20. Wegerer, M., & Tjoa, S. Defeating the Database Adversary Using Deception – A MySQL Database Honeypot. *International Conference on Software Security and Assurance (ICSSA)*, Saint Pölten. Austria, 2016. pp. 6-10. DOI: 10.1109/ICSSA.2016.8.

21. Kedrowitsch, A., Danfeng, Y., Gang. W., & Cameron, K. A First Look: Using Linux Containers for Deceptive Honeypots. *Proceedings of the 2017 Workshop on Automated Decision Making for Active Cyber Defense (SafeConfig '17)*. Association for Computing Machinery, New York, NY, USA, 2017, pp. 15–22. DOI: 10.1145/3140368.3140371.

22. Almeshekah, M. H., & Spafford, E. H. Cyber Security Deception. In: Jajodia. S., Subrahmanian. V., Swarup. V., Wang. C. (eds). *Cyber Deception*, 2016, p. 318, Cham. Springer. DOI: 10.1007/978-3-319-32699-3_2.

23. Zobal, L., Kolář, D., & Fujdiak, R. Current State of Honeypots and Deception Strategies in Cybersecurity. *11th International Congress on Ultra-Modern Telecommunications and Control Systems and Workshops (ICUMT)*. Dublin. Ireland. 2019. pp. 1-9. DOI: 10.1109/ICUMT48472.2019.8970921.

24. Dahbul, R. N., Lim C., & Purnama. J. Enhancing honeypot deception capability through network service fingerprint. *Journal of Physics: Conference Series*, 2017, vol. 801, article no. 012057. DOI: 10.1088/1742-6596/801/1/012057.

25. Razali, M. F., Razali, M. N., Mansor, F. Z., Muruti, G., & Jamil, N. IoT Honeypot: A Review from Researcher's Perspective. *IEEE Conference on Application. Information and Network Security (AINS).* Langkawi. Malaysia, 2018. pp. 93-98. DOI: 10.1109/AINS.2018.8631494.

26. La, Q. D., Quek, T. Q. S., Lee, J., & Zhu, H. Deceptive Attack and Defense Game. Honeypot-Enabled Networks for the Internet of Things. *IEEE Internet of Things Journal*, 2016, vol. 3, no. 6. pp. 1025-1035. DOI: 10.1109/JIOT.2016.2547994.

27. Rowe, N. C. Honeypot Deception Tactics. In: Al-Shaer, E., Wei, J., Hamlen, K., Wang, C. (eds) *Autonomous Cyber Deception*. Springer. Cham, 2019. DOI: 10.1007/978-3-030-02110-8_3.

28. Lysenko, S., Savenko, O., Bobrovnikova, K., & Kryshchuk, A. Self-adaptive system for the corporate area network resilience in the presence of botnet cyberattacks. *Communications in Computer and Information Science*, 2018, vol. 860, pp. 385-401. DOI: 10.1007/978-3-319-92459-5_31.

29. Pomorova, O., Savenko, O., Lysenko, S., Kryshchuk, A., & Bobrovnikova, K. A Technique for the Botnet Detection Based on DNS-Traffic Analysis. *Computer Networks. CN 2015. Communications in Computer and Information Science*, 2015, vol. 522, pp. 127-138. DOI: 10.1007/978-3-319-19419-6_12.

30. Bobrovnikova, K., Lysenko, S., Savenko, B., Gaj, P., & Savenko, O. Technique for IoT malware detection based on control flow graph analysis. *Radioelectronic and Computer Systems*, 2022, vol. 1, pp. 141–153. DOI: 10.32620/reks.2022.1.11.

31. Lysenko, S., Savenko, O., Bobrovnikova, K., Kryshchuk, A., & Savenko, B. Information technology for botnets detection based on their behaviour in the corporate area network. *Communications in Computer and Information Science*, 2017, vol. 718, pp. 166–181. DOI: 10.1007/978-3-319-59767-6_14.

32. Moskalenko, V., Zarets'kyy, M., Moskalenko, A., Kudryavtsev, A., & Semashko, V. Multi-layer model and training method for malware traffic detection based on decision tree ensemble. *Radioelectronic and Computer Systems*, 2020, vol. 2, pp. 92-101. DOI: 10.32620/reks.2020.2.08.

33. Morozova, O., Nicheporuk, A, Tetskyi, A., & Tkachov, V. Methods and technologies for ensuring cybersecurity of industrial and web-oriented systems and networks. *Radioelectronic and Computer Systems*, 2021, vol. 4, pp. 145-156. DOI: 10.32620/reks.2021.4.12.

34. Dovbysh A., Liubchak, V., Shelehov, I., Simonovskiy, J., & Tenytska, A. Information-extreme machine learning of a cyber attack detection system. *Radioelectronic and Computer Systems*. 2022, vol. 3, pp. 121-131. DOI: 10.32620/reks.2022.3.09.

35. Fursov, I., Yamkovyi, K., & Shmatko, O. Smart Grid and wind generators: an overview of cyber threats and vulnerabilities of power supply networks. *Radioelectronic and Computer Systems*, 2022, vol. 4. pp. 50-63. DOI: 10.32620/reks.2022.4.04.

36. Ahmed, J., Karpenko, A., Tarasyuk, O., Gorbenko, A., & Sheikh-Akbari, A. Consistency issue and related trade-offs in distributed replicated systems and databases: a review. *Radioelectronic and Computer Systems*, 2023, vol. 2. pp. 171-179. DOI: 10.32620/reks.2023.2.14.

37. Alnajim, A. M., Habib, S., Islam, M., Albelaihi, R, & Alabdulatif, A. Mitigating the Risks of Malware Attacks with Deep Learning Techniques. *Electronics*, 2023, vol. 12, iss. 14. pp. 3166. DOI: 10.3390/electronics12143166.

38. da Silva, A. A., & Pamplona Segundo, M. On Deceiving Malware Classification with Section Injection. *Machine Learning and Knowledge Extraction*, 2023, vol. 5, iss. 1. pp. 144-168. DOI: 10.3390/make5010009.

39. Saminathan, K., Mulka, S. T. R., Damodharan, S., Maheswar, R., & Lorincz, J. An Artificial Neural Network Autoencoder for Insider Cyber Security Threat Detection. *Future Internet*. 2023, vol. 15, iss. 12, article no. 373. DOI: 10.3390/fi15120373.

40. Markoulidakis, I., Rallis, I., Georgoulas, I., Kopsiaftis, G., Doulamis, A., & Doulamis, N. Multiclass Confusion Matrix Reduction Method and Its Application on Net Promoter Score Classification Problem. *Technologies*, 2021, vol. 9. DOI: 10.3390/technologies9040081.

41. Tharwat, A. Classification assessment methods. *Applied Computing and Informatics*, 2021, vol. 17, no. 1, pp. 168-192. DOI: 10.1016/j.aci.2018.08.003.

42. Powers, D. Evaluation: From Precision. Recall and F-Measure to ROC. *Informedness. Markedness & Correlation*. arXiv 2020. DOI: 10.48550/arXiv.2010.16061.

43. Markoulidakis, I., Rallis, I., Georgoulas, I., Kopsiaftis, G., Doulamis, A., & Doulamis, N. A Machine Learning Based Classification Method for Customer Experience Survey Analysis. *Technologies*, 2020, vol. 8, article no. 76. DOI: 10.3390/technologies8040076.

44. Lysenko, S., Savenko, O., & Bobrovnikova, K. DDoS Botnet Detection Technique Based on the Use of the Semi-Supervised Fuzzy c-Means Clustering. *CEUR-WS*, 2018, vol. 2104, pp. 688-695.

45. Lysenko, S., Bobrovnikova, K., Shchuka, R., & Savenko, O. A Cyberattacks Detection Technique Based on Evolutionary Algorithms. *11th International Conference on Dependable Systems. Services and Technologies (DESSERT)*, 2020, vol. 1, pp. 127-132. DOI: 10.1109/DESSERT50317.2020.9125016.

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

Received 17.07.2023. Accepted 20.11.2023

ПРИНЦИП І МЕТОД СИНТЕЗУ СИСТЕМ ОБМАНУ ДЛЯ ВИЯВЛЕННЯ ЗЛОВМИСНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ І КОМП'ЮТЕРНИХ АТАК

Антоніна Каштальян, Сергій Лисенко, Богдан Савенко, Томаш Сочор, Тетяна Кисіль

Кількість різних типів та безпосередньо сама кількість зловмисного програмного забезпечення і комп'ютерних атак постійно збільшуються. Тому, виявлення та протидія зловмисному програмному забезпеченню та комп'ютерним атакам залишаються актуальною проблемою сьогодення. Особливо найбільшої шкоди зазнають користувачі корпоративних мереж. Для виявлення та протидії їм розроблено багато ефективних засобів різноманітного спрямування. Але динамічність в розробці нового зловмисного програмного забезпечення та урізноманітнення проведення комп'ютерних атак спонукають розробників засобів виявлення та протидії постійно вдосконалювати свої засоби та створювати нові. Об'єктом дослідження в роботі є системи обману. Результати цієї роботи розвивають елементи теорії та практики створення таких систем. Особливе місце серед засобів виявлення та протидія зловмисному програмному забезпеченню та комп'ютерним атакам займають системи обману. Ці системи заплутують зловмисників, але теж потребують постійних змін та оновлень. оскільки з часом особливості їх функціонування стають відомими. Тому, актуальною є проблема створення систем обману, функціонування яких залишалось би незрозумілим для зловмисників. Для вирішення цієї проблеми в роботі пропонується но-вий принцип синтезу таких систем. Оскільки формування таких систем буде на базі комп'ютерних станцій корпоративної мережі, тоді систему позиціоновано як мультикомп'ютерну. В системі запропоновано використовувати комбіновані приманки та пастки для створення хибних об'єктів атак. Всі компоненти такої системи формують тіньову комп'ютерну мережу. В роботі розроблено принцип синтезу мультикомп'ютерних систем з комбінованими приманками і пастками та контролером прийняття рішень для виявлення та протидії зловмисному програмному забезпеченню та комп'ютерних атакам. В основу принципу закладено наявність контролера за прийнятими в системі рішеннями та використання спеціалізованого функціоналу з виявлення та протидії. Згідно розробленого принципу синтезу таких систем в роботі виділено підмножину систем з технологіями обману, в яких обов'язково повинен бути контролер та спеціалізований функціонал. Контролер за прийнятими рішеннями в системі є відокремленим від центру прийняття рішень. Його завданням є вибір варіантів наступних кроків системи, які сформовані в центрі системи, в залежності від повторюваності подій. Причому тривале повторення зовнішніх подій вимагає від центру системи формування послідовності наступних кроків. За умови їх повторення зловмисник отримує можливість вивчати функціонування системи. Контролер в системі вибирає з різних можливих варіантів відповідей при однакових повторюваних підозрілих подіях різні відповіді. Таким чином, зловмисник при дослідженні корпоративної мережі на одні й ті ж запити отримує різні варіанти відповідей. Спеціалізований функціонал згідно принципу синтезу таких систем імплементовано в архітектуру систем. Він впливає на її зміну архітектури системи в процесі її функціонування в результаті внутрішніх та зовнішніх впливів. В роботі також розглянуто можливий варіант архітектури таких систем обману, зокрема, архітектура системи з частковою централізацією. Для синтезу таких систем розроблено новий метод синтезу частково централізованих систем для виявлення зловмисного програмного забезпечення в комп'ютерних мережах, який базується на розроблених аналітичних виразах, що визначають стан безпеки таких систем та їх компонентів. Проведено експериментальні дослідження розробленої системи на предмет можливості її функціонування тривалий час та виконання поставлених завдань в умовах втрати нею частини компонентів. Результати двох експериментів з п'ятьма серіями підтвердили ефективність запропонованого рішення. Крім того, за результатами експериментів було встановлено, що втрата 10-20% компонентів не впливає на виконання поставленого завдання. Результати експериментів були опрацьовані з використанням ROC-аналізу та алгоритму побудови ROC-кривої. Результати експериментів дали змогу визначити ступінь деградації так побудованих систем.

Ключові слова: системи обману; синтез систем обману; принцип синтезу систем. контролер. розподілені системи; павутина; пастка; приманки; виявлення шкідливого програмного забезпечення; часткова централізація.

Каштальян Антоніна Сергіївна – канд. техн. наук, доц. каф. фізики та електротехніки, докторантка, Хмельницький національний університет, Хмельницький, Україна.

Лисенко Сергій Миколайович – д-р техн. наук, проф., проф. каф. комп'ютерної інженерії та інформаційний систем, Хмельницький національний університет, Хмельницький, Україна.

Савенко Богдан Олегович– асп. каф. комп'ютерної інженерії та інформаційний систем, Хмельницький національний університет, Хмельницький, Україна.

Сочор Томаш – доц. каф. економіки та економічної політики, Університет Пріго, Чеська Республіка.

Кисіль Тетяна Миколаївна – канд. фіз.-мат. наук, доц., доц. каф. комп'ютерної інженерії та інформаційний систем, Хмельницький національний університет, Хмельницький, Україна.

Antonina Kashtalian – PhD, Associate Professor at the Department of Physics and Electrical Engineering, Doctoral Staff, Khmelnytskyi National University, Khmelnytskyi, Ukraine,
e-mail: yantonina@ukr.net, ORCID: 0000-0002-4925-9713.

Sergii Lysenko – Dr. S, Full Professor, Professor at Computer Engineering & Information Systems Department, Khmelnytskyi National University, Khmelnytskyi, Ukraine,
e-mail: sirogyk@ukr.net, ORCID: 0000-0001-7243-8747.

Bohdan Savenko – PhD Student at Computer Engineering & Information Systems Department, Khmelnytskyi National University, Khmelnytskyi, Ukraine,
e-mail: savenko_bohdan@ukr.net, ORCID: 0000-0001-5647-9979.

Tomáš Sochor – Associated Professor for Cybersecurity and Quantitative Methods, Department of Economics and Economic Policies, Prigo University, Czech Republic,
e-mail: tomas.sochor@prigo.cz, ORCID: 0000-0002-1704-1883.

Tetiana Kysil – PhD, Associate Professor at the Department of Computer Engineering & Information Systems Department, Khmelnytskyi National University, Khmelnytskyi, Ukraine,
e-mail: kysil_tanya@ukr.net, ORCID: 0000-0002-4094-3500.