The Hardware and Software Part of a Distributed Decentralized Systems Detection Malicious Software

Oleg Savenko [a], Mykola Stetsyuk [a], Yuriy Stetsyuk [a], Antonina Kashtalian[a], Bohdan Savenko [a]

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

Abstract

The paper analyzes the problems that arise from ensuring the security and protection of information in corporate networks of enterprises. A variety of tools are used to detect malware in the computer of the power grids of enterprises. This approach makes it possible to improve the security and protection of information resources of enterprises. For corporate networks of enterprises, such tools are usually distributed. Attackers have knowledge of standard known tools detection of malware in corporate networks, and also have the appropriate means to carry out attacks by various methods. Therefore, to counter them in corporate networks, not only standard sets of network screens, intrusion detection systems, attack prevention systems, anti-virus tools, but also various means should be used. Such tools can baits developed individually for a specific corporate network, malware detection systems. In this case, it will be difficult for the attacker to go unnoticed when trying to invade the corporate network. The use of exclusively anti-malware or malware detection tools affects the effectiveness of the protection process and its effectiveness compared to hardware and software.

The work proposes to improve the security and protection of information in corporate networks to use the individual means of users of the corporate network and, at the same time, they would be part of the malware detection system. These individual means of identifying users in a corporate network would become part of one large sensor at the same time. This sensor would be a decentralized distributed malware detection system. Computational actions would be sent to a certain part of the component, and they, in turn, would use hardware and software to support the operation of the system as a whole and would make calculations in them. If such a hardware and software element of the system components would give an incorrect result compared to the rest, then the system itself would respond to the actions of such a component. This provides the ability to automate the malware detection process and make it harder for an attacker to understand this system.

The results of the experiments with the developed system according to the proposed approach confirm the possibility of their implementation and effective use in corporate networks.

Keywords 1

Distributed systems, USB calculator, schematic diagram, microcontroller, malicious software, honynet

1. Introduction

In order to detect malware in the computer networks of enterprises, a variety of tools are used. Diversifying the tools allows you to improve the security and protection of information resources of enterprises. For corporate networks of enterprises, such tools are usually distributed and they use multi-agent technologies [1].

IntelITSIS'2023: 4rd International Workshop on Intelligent Information Technologies and Systems of Information Security, March 22–24, 2023, Khmelnytskyi, Ukraine

EMAIL: savenko_oleg_st@ukr.net (O. Savenko); mikstt777@gmail.com (M. Stetsyuk); y.stetsuk@ultra-company.com (Yu. Stetsyuk); yantonina@ukr.net (A. Kashtalian); savenko_bohdan@ukr.net (B. Savenko)

ORCID: 0000-0002-4104-745X (O. Savenko); 0000-0003-3875-0416 (M. Stetsyuk); 0000-0003-0312-2276 (Yu. Stetsyuk); 0000-0002-4925-9713 (A. Kashtalian); 0000-0001-5647-9979 (B. Savenko)

Attackers have knowledge of standard known malware detection tools on corporate networks, and also have the appropriate means to carry out attacks using a variety of methods. Therefore, to counter them in corporate networks, not only standard sets of network screens, intrusion detection systems, attack prevention systems, anti-virus tools, but also various means that the attacker is not aware of, should be used. Baits can be used by such means developed individually for a specific corporate network, malware detection systems, which are placed at a certain level among the installed anti-attack systems. That is, in this case, it will be difficult for the attacker to go unnoticed when trying to invade the corporate network.

When using only antimalware or malware detection software, the effectiveness of the protection process and its effectiveness compared to hardware [2] and software is lower. Overcoming the hardware and software protection of corporate networks is a much more difficult task.

An attacker may be in the middle of a corporate network and, even, may have certain rights of access to enterprise resources. Therefore, the use of certain distributed malware detection systems may not be effective enough. This is due to the fact that an attacker can violate at least one of the properties of information, in particular, integrity, accessibility, or confidentiality. As a result, it can distort even unique malware detection tools on the corporate network, which will lead to the concealment of its activities. In connection with In this regard, it is proposed to add hardware and software to such distributed malware detection systems on the corporate network, which would complicate the work of attackers and at the same time improve the security and protection of information on the network.

Such tools could include the individual means of each user of the corporate network and, at the same time, they would be part of the malware detection system. Thus, these individual means of identifying users in a corporate network would become part of one large sensor at the same time. This sensor would be a decentralized distributed malware detection system (DDS) [1]. Computational actions would be sent to a certain part of the component, and they, in turn, would use hardware and software to support the operation of the system as a whole and would make calculations in them. If such a hardware and software element of the system components would give an incorrect result compared to the rest, then the system itself would respond to the actions of such a component. This provides the ability to automate the malware detection process and make it harder for an attacker to understand this system.

2. Analysis of known solutions

Researchers pay a lot of attention to malware detection. This has become especially relevant in recent years and today this area is gaining momentum. As for the means, their architectures, which would be the basis for the synthesis of various means of detection, such a combination in scientific works is not enough. Therefore, the principles of creating distributed architectures that could be used to create distributed malware detection systems in corporate networks are common, that is, without taking into account the peculiarities of such detection tools.

Consider the following features in scientific works. The results of scientific work are mainly elements of distributed systems, their protocols of interaction of components, principles of construction. Consider them in this aspect. In the works [3-4] Byzantine protocols are analyzed and their synchronization is proposed to ensure the result. In the works [5-7] an autonomous distributed system for heterogeneous storage of information is proposed. The system has developed different levels of storage. In the works [8-9], experimentally evaluating the method of comparing the scheme based on instances for attributes that store numerical data. It uses data distribution and correlation between two attributes. Working with databases is important in the context of distributed processing. In [10] the paper considers the cyber-physical production system for the relationship between operational technology and information and communication technology between machines and decentralized production management. The architecture of this system is decentralized and, according to the principle of construction, has common criteria by which DDS is synthesized. The work [11] deals with security incidents in cyber-physical systems. A new approach is proposed to present and share knowledge about incidents between different organizations. In [12] the principles of creating distributed systems are considered. In [13] considered distributed computing environment.

An equally important direction that should be taken into account when projecting DDS is to take into account malware detection methods that can be based on evolutionary algorithms, fuzzy inference systems, etc. [14-17]. And these methods should be organically combined with the architecture of the system and its component. In [18] analyzed security incidents that can be in distributed systems. Work 19] analyzes the protection of hardware components of computers. In [20-24] the analysis of trends in the development of malware and means of counteracting them is carried out. In [25] the methods of malware analysis are considered. Manuscripts [26-30] are devoted to malware detection.

3. Hardware and software part of a component of a distributed decentralized systems

To improve security when using a distributed decentralized malware detection system, a USB computer must be added to its software implementation.

A USB computer is a hardware and software part of a component of a distributed decentralized malware detection system and is designed to perform calculations. As part of its block diagram shown in Fig. 1 contains a computing unit, non-volatile memory of programs and data, an operational storage device and a USB interface controller, combined into a single circuit with the corresponding buses.

Consider thein scription of the functional scheme of the USB-calculator. A USB computer is designed to perform calculations in the interests of a decentralized distributed system for detecting malware.

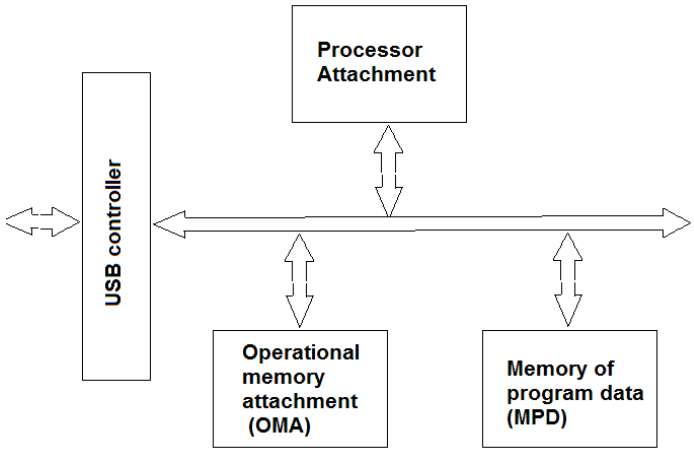

Figure 1: Functional diagram of USB – calculation

The computing device is designed to directly perform calculations to determine the security status of a decentralized distributed system by commands that are received from the host, which includes a USB computer, to save the results of calculations in the data memory, followed by their transfer at the request of the host. Its functional scheme (Fig. 1) includes a processor device, non-volatile memory of programs and data, RAM and a USB interface controller, combined into a single scheme by the corresponding buses.

The processor device is designed to directly perform calculations for determining the security status of a DDS by commands received from the host, which includes a USB computer, saving the results of calculations in data memory and subsequently transmitting them at the request of the host.

Program and data memory (PPD) is used to store programs that control the operation of the USB computer, save the results of calculations of the state of the DDS.

RAM is designed to store current local data used during the computational process.

The USB interface controller serves to enable data exchange between the USB computer and the host.

Consider the description of the schematic diagram of a USB computer. The developed schematic diagram (Fig. 2) is a variant of the implementation of the USB computer diagram described above. Its

feature is the implementation in the form of a micromodule with overall dimensions corresponding to the flash drive.

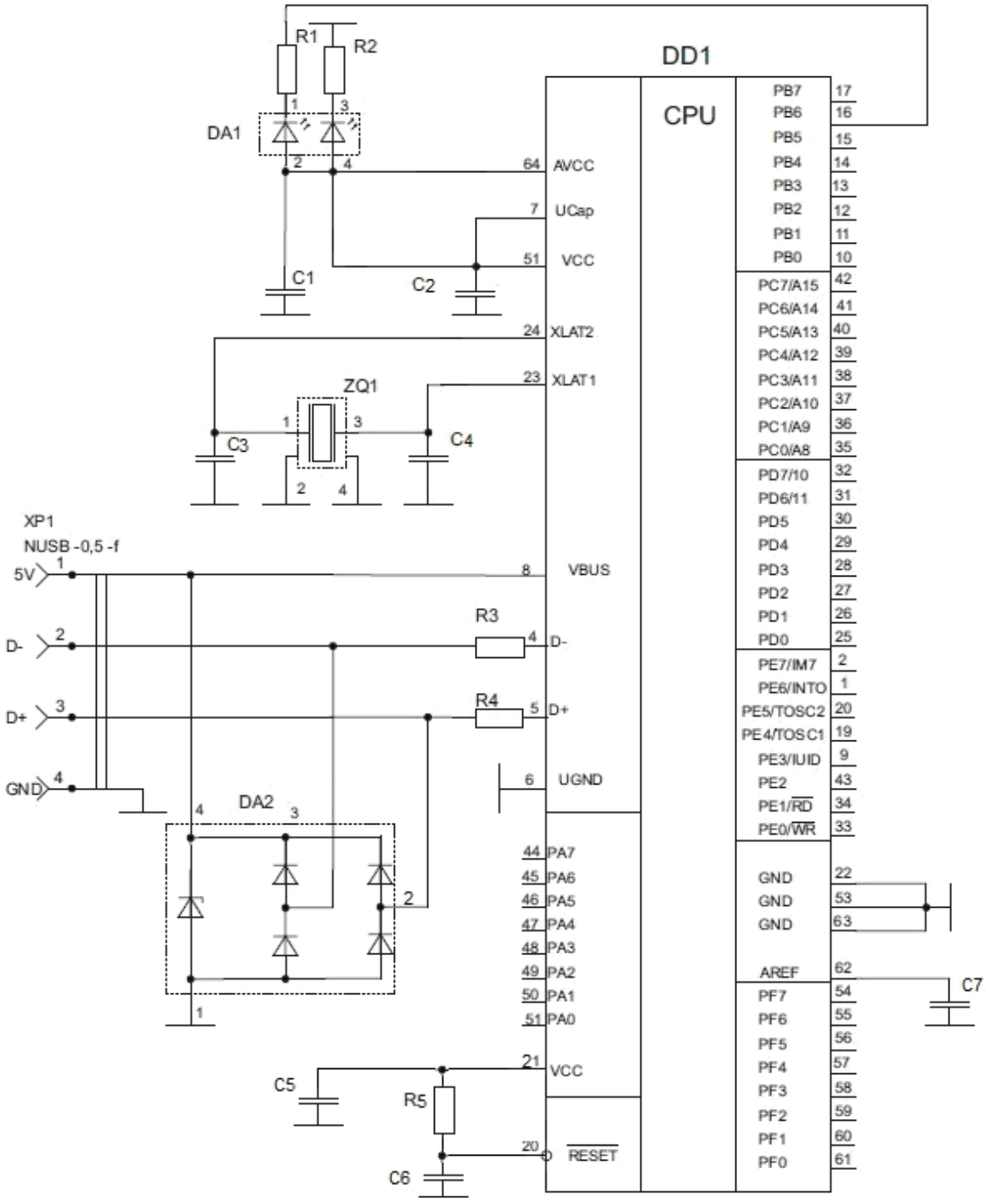

Figure 2: Electrical circuit diagram

The basis of the circuit is the chip DD 1 - the microcontroller of the company Atmel AT90USB1287-16MV. This is an 8-r KMOP-m and a controller, on the bases and archandtecturи AVR RISC. By score of execution of b and more commands for one pen and od synchrony and produkyouvnity of this microcontroller dos one m and l and wan operation and even per second at 1 MHzclock speed i. Its AVR is a core set of 135 commands that can manipulate 32 universal working registers. This allows you to implement in the code of this microcontroller quite complex calculation algorithms DDS.

Table 1

USB principal computer circuit specification

Denomination	Name	Quantity
	Chip	
DD1	AT90USB1287-16MU	1
DA1	SML-020MLT	1
DA2	PRTR5V0U2X	1
ZQ1	8Mhz, GSX-752	1
	Resistors	
R1,R2	SMD 0201 0,05Вт 220 Ом +-5%	2
R3,R4	SMD 0201 0,05Вт 22 Ом +-5%	2
R5	SMD 0201 0,05Вт 100 кОм +-5%	1
	Capacitors	
C1,C5-C7	SMD 1206 X7R 50V 0,1мкФ +-10%	3
C2	SMD 1210 X7R 50V 1,0 мкФ +-10%	1
C3,C4	SMD 1206 NPO 50V 18 пФ +-5%	2
	Connectors	
XP1	NUSB-0,5-f	1

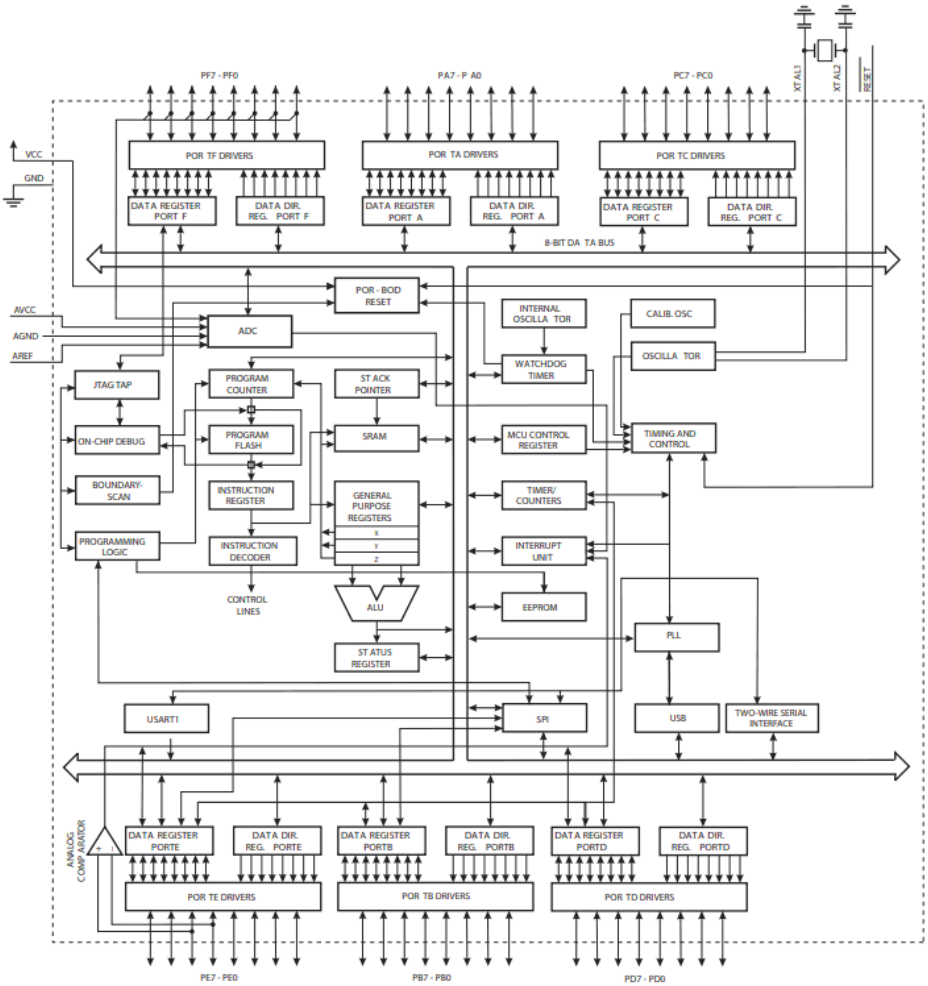

Figure 3: Structural diagram of the microcontroller AT90USB1287

In addition to the productive arithmetic-logical device, the architecture of the microcontroller (Fig. 3) includes 128 kbytes of non-volatile, electrically programmable flush memory with read-write support, 4 KB of software ROM, 4 KB of RAM. This made it possible to implement the PPD and RAM modules of the functional scheme internally systematically, without including additional components in the schematic diagram.

The architecture of the AT90 USB1287-16MV microcontroller (Fig. 3) includes a hardware implementation of the USB data transfer interface (Fig. 4). This not only allowed the internal system to implement the USB controller of the functional circuit, but also to greatly simplify the procedure for initializing the USB computer and abandon the use of the programmer.

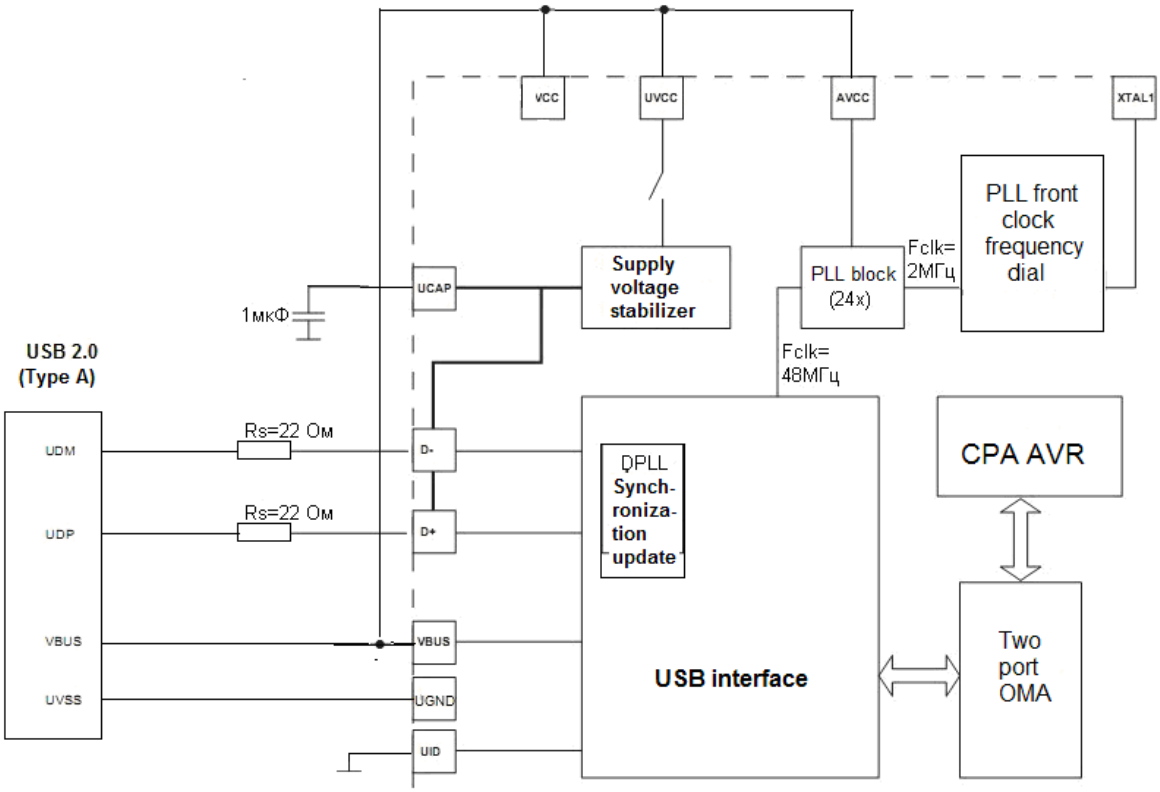

Figure 4: Block diagram of the USB controller and the power supply circuit of the AT90USB1287 mikrocontroller from the VBUS bus of the USB port

The USB controller contains all the necessary components to connect the USB channel to the built-in dual-port RAM (DPRAM). It is synchronized with a frequency of 48 MHz±0.25% (for operation in FS mode), which is generated by the PLL unit. Its peculiarity is that the high-frequency signal (48 MHz) is synthesized from a low-frequency signal (2 MHz). The source of this signal is the quartz generator ZQ 1, connected to the input XTAL 1 of the DD1 chip and the previous frequency divider of the synchronization unit of the PLL. This method of receiving a 48MHz synchronization signal made it possible to meet the requirements of the USB controller for frequency stability and phase noise, which ensures its proper functioning.

The 48 MHz synchronization signal is further used to generate a bit synchronization signal with a frequent 12 MHz in FS mode (or 1.5 MHz in LS mode) when receiving and transmitting differential data, taking into account permissible deviations in the corresponding speed mode. The resumption of synchronization is performed by the block of digital phase auto-tuning of the frequency (DPLL unit).

To meet the electrical performance requirements of the USB bus, the D+ and D- outputs must have high voltage levels in the 3.0... 3.6V. For this purpose, the microcontroller has a built-in special voltage stabilizer, which made it possible to use its power supply circuit with voltages up to 5.5V.

In addition to the hardware internally present in the DD 1 chip, current limiting resistors R3 and R4, the DA2 PRTR5VU2 chip are included in the principle diagram of the USB controller. Together with the resistors R 3 and R4, it performs the protective and stabilizing functions of the USB controller.

The DA1 chip contains two diode lights, which serve as indicators of power supply to the USB computer and the access to it of the host in which it is used.

The inclusion of the indicator for accessing the USB-computer is performed programmatically by writing the value of logical zero to the sixth digit of port B.

As DA1, considering the overall dimensions as a priority parameter, the SML-020MLT microassembly was selected.

The resistances R1 and R2 perform the function of limiting the current that passes through the light of the diodes of the microassembly DA1.

The ZQ1 element connected to the XLAT1 and XLAT2 DD1 inputs serves to excite the microcontroller synchronization system. It sets the operating frequency of the clock pulse generator at 8 MHz. Its standard connection requires inclusion in the princely circuit of capacitors C3, C4. As ZQ1, a small-sized component G5K-752 was used.

Capacitors C1, C2, C 5, C6 implement the standard, recommended by the developer, connection scheme of the microcontroller, to ensure the appropriate quality of the voltage of the sameequalization.

Resistor R5 and capacitor C6 serve to generate the hardware signal for installing the USB computer circuit to its original state, which is fed to the inverse input DD1 RESET. The duration of the RESET signal depends on the nominal values of the resistor R5 and the capacitor C6 and for reliable installation of the CPU DD1 in RESET (Fig. 8), 32 synchronization cycles are required.

When the USB computer is connected to the USB port of the host, a logical zero signal is set at the input D 1 RESET, since the C6 capacitor is currently discharged and the potential of its top cover is zero. Next, the C6 capacitor begins to charge through the resistor R5 and, accordingly, the potential of its top cover begins to grow. When it reaches the level of a logical unit, the RESET signal stops and the CPU chip DD1 moves from the RESET state to the IDLE state and then to the DEVICE state to execute the first command of program module 2.

As an XP1 connector, a USB connector MUSB-0.5-Type A pin 90 was used, which has minimal mounting dimensions due to the placement of outputs at an angle of 90 degrees.

The assembled circuit does not require hardware debugging and is ready for the initialization procedure.

Let's initiate a USB calculator on the host side. USB calculator, like any other USB device, supports "hot" (plug'n'play) connection with dynamic loading and unloading of drivers. After the user inserts the device into the USB port, host finds this when connected, polls t and but in thein stalled device and loads the appropriate driver. This USB computer is defined in Windows as a device AT90USB1287 (Fig. 5).

Figure 5: Devices window of the Windows operating system

To do this, do not bypass the program Flip installer from Atmel of the corresponding version. It must be installed on the computer on which the USB computer will be initialized.

When you first connect the USB computer via a USB port, the system will ask for its driver. You must specify the path "c:\Program Files\Atmel\Flip 3. x. x\usb\ “. There is a driver for the bootloader a loader and Flip installer.

After completing the sequence of steps described above, the device AT90USB 1287 will appear in the system, as can be seen in the figure. 5.

We will directly initiate the USB-calculator.

Initialization of the USB computer is performed by loading the application software (firmware) into the memory of the programs of the AT90USB1287 microcontroller when it is prepared for work as part of DDS.

To do this, you need to use the software of Atmel, the developer of the AT90USB1287 microcontroller, which allows you to program the microcontroller memory by command via a USB port via the Atmel® USB DFU protocol**.**

In the microcontroller DD1 (Fig. 2), when it is delivered, there is a flashed a programs bootloader, another name is Device Firmware Uploader (DFU), which serves to load the first and second software modules of the USB computer into memory controll era via USB -and interface her on thei konnya.

In this case, the second module is loaded from the address 0000H. This is the starting address of the processor device of the microcontroller. After that, when the USB computer is connected to the host, each time the hardware initialization procedure will be performed (setting the microcontroller circuits to its original state) (*described in paragraph 5 of paragraph 8*) of the USB computer, as a result of which control will be transferred to the first command of the second software module.

Let's set the algorithms for the operation of the software (software) of the USB-calculator.

The algorithms of the USB computer software are built on the basis of the rule that the initiator of the transaction can be a host machine, which includes a USB computer.

The application software of the USB calculator consists of two main modules that interact through a common part of the RAM, used as a ring software command buffer (CBC) of 256 bytes (Fig. 6).

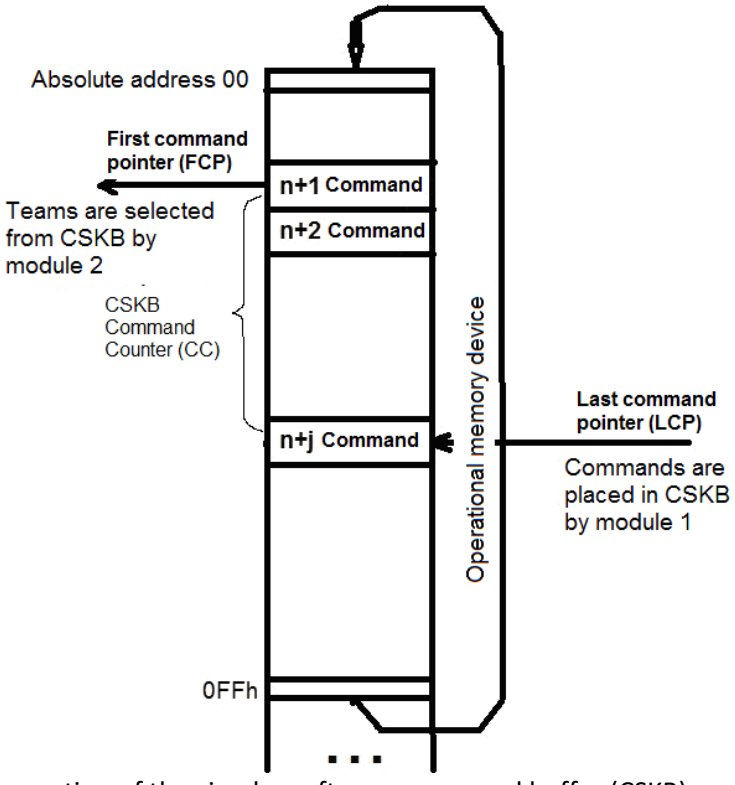

Figure 6: Scheme of operation of the circular software command buffer (CSKB)

Since computational processes performed by the host machine and USB computer are asynchronous relative to each other, therefore, the use of an annular buffer allows you to organize a channel for transmitting commands from the host to the USB computer at the lowest cost.

The first software module (Fig. 7) is implemented as a subroutine for processing interrupts of the microcontroller processor and is responsible for receiving commands from the host under the control of DDS to perform calculations of its security status, storing the results of these calculations and issuing them at the request of DDS components.

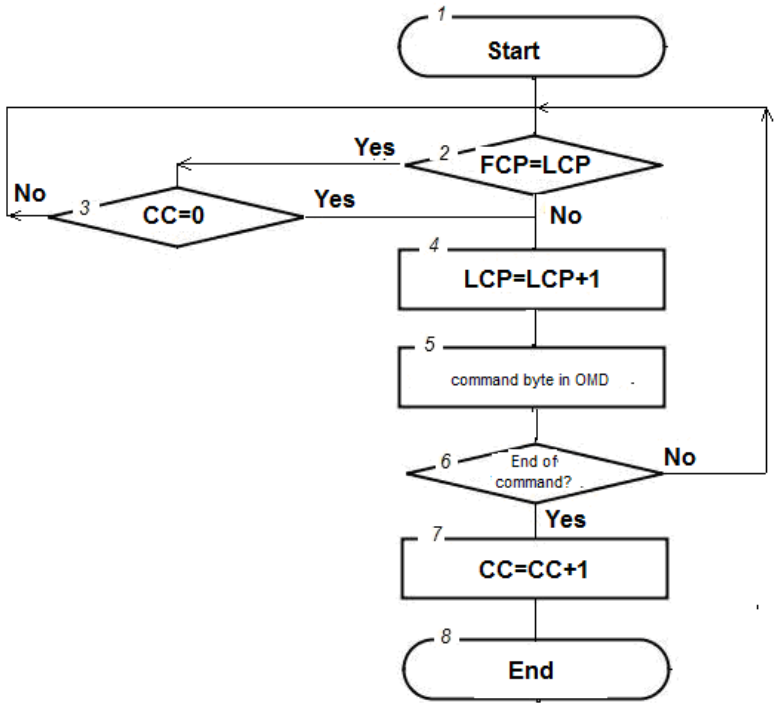

Figure 7: Module 1 operation algorithm

This module is activated by interruption signals for USB controller transactions. It is entrusted with the control of the CPBC USB-computer. If the buffer is not full of commands, then the module places the commands received via the USB interface in the command buffer with simultaneous modification of the pointer to the command (WOK).

Consider the steps of the algorithm from Fig. 7.

1.Start of the procedure for receiving the next command from DDS (module 1).

2.Checking the filling of the loop buffer of DDS commands. If the pointer of the last LCP command does not match the pointer of the first FCP command contained in the buffer, then go to step 4.

3.Check the value of the CC command counter. If its value is bandmore than zero (the buffer is completely filled with commands), then we go to step 2.

4. Move the LCP pointer to the next free cell of the command buffer.

5. Take the next byte of the command from DDS and place it at the address indicated by the LCP pointer.

6. Check whether the last byte of the command is accepted. If not, then we go to step 2.

7. Increase by one the value of the CC command counter.

8. End of the procedure.

The operation of the USB computer software is organized in such a way that when it is connected to the host, its hardware is installed in its original state, after which the processor starts, the first command of which is the command that is part of the second software module. Thus, the control of the USB computer immediately after the start is transferred to the second software module.

The second software module is a USB-computer application program (Fig. 9).

After passing the RESET hardware signal, the microcontroller is installedin the initial idle state (Fig. 8).

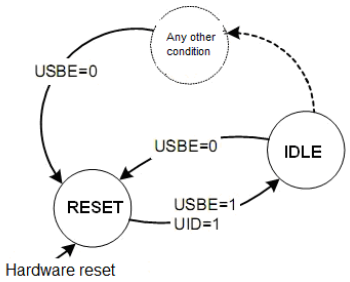

Figure 8: Grapf of microcontroller states

In this mode, the processor core enters the stop mode. IDLE mode does not affect the activity of the USB controller: it may and may not work. The processor core is output to the active state on the first interruption from the USB controller.

When the USB computer is connected to the USB port of the host, it will detect the connection of the new device. At the same time, on the side of the USB calculator, when power is supplied to the device, a voltage appears on the VBUS bus, which leads to setting the DETACH bit to a logical zero state, which in turn turns on the tightening to plus D+ signals or to zero D- and thereby initiates a data channel between the host and the USB controller.

When the host system contacts it in order to determine its parameters, an interrupt signal from the USB controller will appear on the USB computer side, which will transfer the core of the microcontroller processor to the active state.

This, in turn, will lead to the execution of the code of the software module 2, which begins with the procedure for initializing the microcontroller in DEVICE mode (Fig. 9 block 2) and involves the following steps:

- configuring and activating endpoint 1 to receive data from the host;
-configuringand activating endpoint 2 to transfer data to the host;
-selection of LS speed mode (bit LSM=1 in VDCON register).

The next step is the initialization of the CPBC with the allocation of memory for it (Fig. 9 block 3), setting the initial values of the pointers of the first and last commands, issuing permission to interrupt the processor device and, thereby, allowing the operation of the first software module.

After completing the initialization procedure, the software module two enters the mode of monitoring the state of the CPBC (Fig. 9 block 5).

If the CPBC is not empty, then the module reads the command using the first command pointer (MIC) with the subsequent modification of the military-industrial complex (fig. 9 block 7).

In the next step, the RBC team selected from the CPBC is decrypted (blocks 12-14,18,19). The DDS command system for the USB computer includes five basic commands K01 - K05 and can be expanded.

The K01 command is designed to clean the non-volatile memory of the USB calculator, allocated for saving the results of DDS security state calculations.

The K02 and K05 commands are designed to perform calculations of the security states of the DDS at the 1st and 2nd stages, respectively.

Team K02 (fig. 9 block 1 6) programmatically implements the algorithm for calculating the state DDS according to the formula (1):

$$P_{DDS,2} = \frac{\sum_{j=1}^{n}\sum_{\substack{s=1,\\ t'_{\Pi M,s,j}>0,\\ w'_{\Pi M,s,j}>0}}^{8} (t'_{\Pi M,s,j} \cdot w'_{\Pi M,s,j})}{n}, \quad (1)$$

where is the probability of being affected for the DDS $P_{DDS,2}$ based on the probabilities of the time-generated stay in certain states and the number of stays in the context of the DDS program modules.

The calculation of the product for a certain state is carried out provided that the components has

been in it at least once. If the components was in a certain state at least once, then the time of its stay will be more than zero, what is needed to obtain the value of the term. The denominator contains a number equal to the number of active components in the DDS at the current time. Since components are active, then they have indicators of time and amount of stay in certain states, the values of which are not zero.

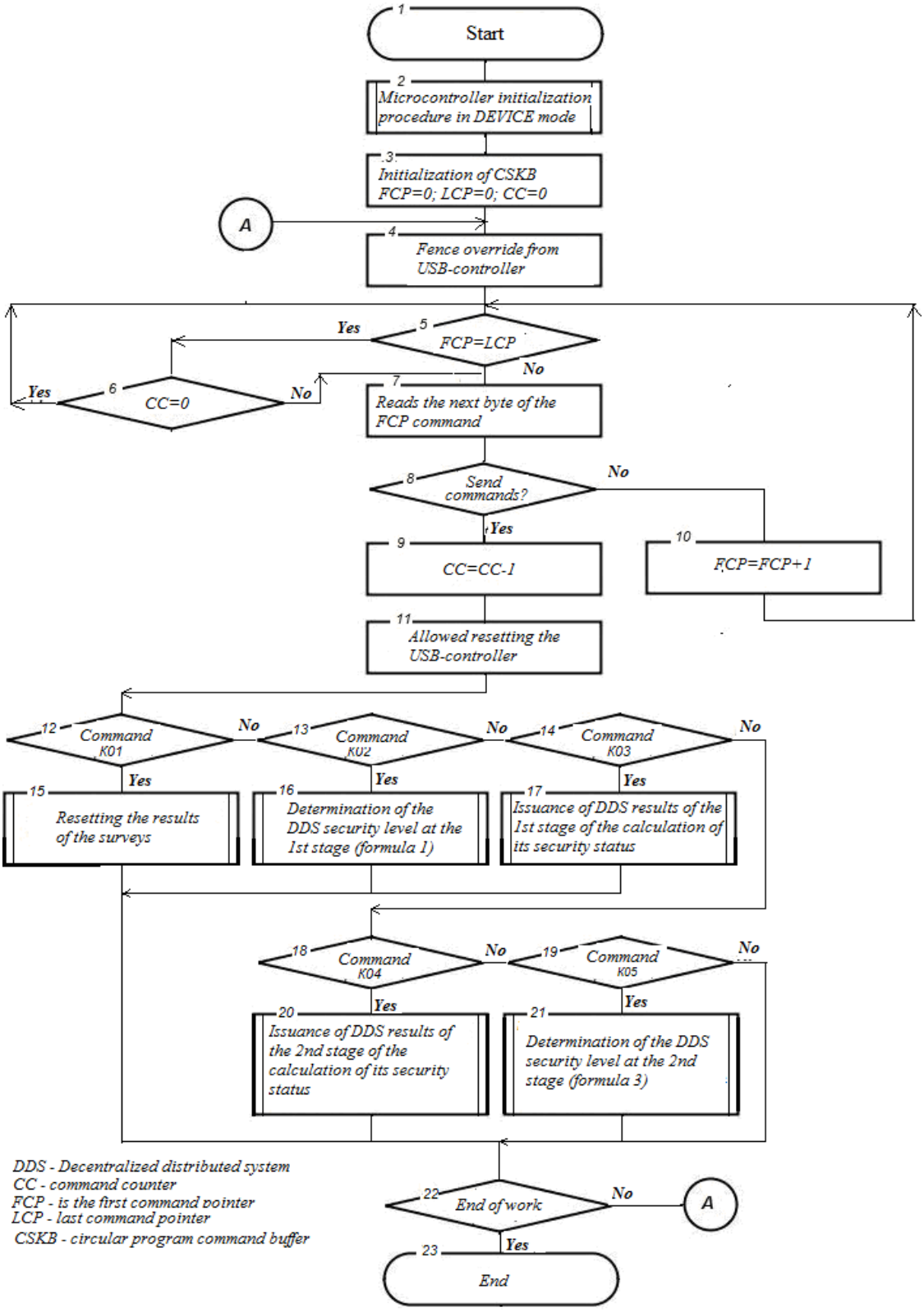

Figure 9: Algorithm of the 2nd software module

Command K05 (fig. 9 block 21) according to the formula (2):

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

where $R_{b,DDS,2}$ is the level of security of the DDS, determined at the second stage; b – security designation; s – number of the DDS program module; n – the number of DDS software modules; m – the number of components states; – the k_s coefficient of threat to be affected by the components – the state of the components, the value of which is set from the segment $[0;1]$, depending on what functional loads are laid down in a certain s state; $p_{s,j}$ – the being affected by the DDS; $w_{s,j}$ - the number of stays of the components with the number j in the states; $i = 1,2,\dots,n$; $s = 1,2,\dots,m$; – the $t_{s,j}$ total time of stay of the components with the number in the state j – the number of components.

Commands K03 and K04 serve to issue DDS results for calculating its safety states at stages 1 and 2, respectively. After executing the current command, the software module returns to the CPBC status monitoring point.

Description of steps to fig. 9.

1. Start of the application program for executing the command of the DDS (module 2).
2. Performing the initialization procedure of the microcontroller in DEVICE mode.
3. Performing the initialization procedure of the CSKB ring software buffer (CC=0; FCP=0; LCP=0).
4. Issuance for interruption of the microcontroller (Permission to perform the procedure of module 1).
5. Checking the filling of the cyclic DDS command buffer. If the pointer of the last LCP command does not match the pointer of the first FCP command contained in the buffer, then proceed to step 7.
6. Check the value of the CC command counter. If its value is zeroy (the buffer is empty), then we go to step 5 (we are waiting for the command to arrive).
7. Reading the next byte of the command from DDS.
8. Check if this is the last byte of the DDS command. If not, then we go to step 10.
9. Reduce the contents of the CC command counter, after completing the sample from the ring buffer of the next DDS command.
10. Move the pointer of the first FCP command to the next one contained in the buffer. Go to step 5.
11. Prohibit external interrupts for the duration of execution of the DDS command selected from the buffer.
12. If this is command K01, then we perform the transition to step 15.
13. If this is command K02, then we perform the transition to step 16.
14. If this is command K03, then we perform the transition to point 17, if not, then to paragraph 18.
15. Execution of command K01. The transition to point 22 is completed.
16. Execution of the K02 command. The transition to point 22 is completed.
17. Execution of command K02. The transition to point 22 is completed.
18. If this is command K04, then we perform the transition to step 20.
19. If this is command K05, then we perform the transition to step 21.
20. Execution of the K04 command. The transition to point 22 is completed.
21. Execution of the K05 command. The transition to point 22 is completed.

22. If the program for executing DDS commands is not the end of the program, then we proceed to step 4.

23. End of work.

Thus, the hardware and software element of the DDS components has been developed. Its use by users of corporate networks will improve the security and protection of information in it.

4. The results of experimental studies with DDS, in which the hardware and software elements of the component are installed

Entries from the DDS log file that display the results of the system for several hours.

WorkingState: 04.02.2023 10:17:05.064 Scheduling next task. Working period = 10 minutes.

WorkingState: 04.02.2023 10:17:05.065 Start working on task: Checking running processes

WorkingState: 04.02.2023 10:17:05.067 Complete work on task: Checking running processes, results: running processes = 5

WorkingState: 04.02.2023 10:27:05.104 Scheduling next task. Working period = 10 minutes.

WorkingState: 04.02.2023 10:27:05.104 Start working on task: Checking files in hard disk in C:/Users/8.1x64/AppData/Local

WorkingState: 04.02.2023 10:27:05.337 Complete work on task: Checking files in hard disk in C:/Users/8.1x64/AppData/Local, results: total dirs count = 802, total files count = 835, total exe files = 16

WorkingState: 04.02.2023 10:27:05.337 Start working on task: Scanning files in hard disk

WorkingState: 04.02.2023 10:27:05.337 Complete work on task: Scanning files in hard disk, results: scanning files = 16, infected files = 0

WorkingState: 04.02.2023 10:36:05.572 169.254.169.70 started working on: Optimizing stored data

WorkingState: 04.02.2023 10:36:05.572 169.254.169.70 completed working on task: Optimizing stored data, results: deleted entries = 11, file size before = 8418, file size after = 7335

WorkingState: 04.02.2023 10:36:05.588 169.254.169.70 started working on: Waiting next task

WorkingState: 04.02.2023 13:48:12.178 Scheduling next task. Working period = 116 minutes.

…

WorkingState: 04.02.2023 14:12:15.103 Start working on task: Optimizing stored data

Message: 04.02.2023 14:12:15.134 Send message about completing working on: Optimizing stored data

WorkingState: 04.02.2023 14:12:15.134 Complete work on task: Optimizing stored data, results: deleted entries = 37, file size before = 15350, file size after = 11985

Message: 04.02.2023 14:12:15.134 Send message about starting woking on: Waiting next task

Message: 04.02.2023 14:12:17.197 Receive startup message from 169.254.169.70

Message: 04.02.2023 14:12:17.197 Send greeting to 169.254.169.70

Message: 04.02.2023 14:12:17.212 Receive poll out message from 169.254.169.70

Message: 04.02.2023 14:12:17.212 Send current state to 169.254.169.70

WorkingState: 04.02.2023 14:12:19.212 169.254.169.70 started working on: Optimizing stored data

WorkingState: 04.02.2023 14:12:19.228 169.254.169.70 completed working on task: Optimizing stored data, results: deleted entries = 15, file size before = 11893, file size after = 10763

WorkingState: 04.02.2023 14:12:19.228 169.254.169.70 started working on: Waiting next task

Message: 04.02.2023 14:12:51.962 Send shutdown message
Message: 04.02.2023 14:12:51.962 Engine is deinitialized

The obtained DDS results confirm the possibility of such implementation of the hardware and software elements of its components.

The results of the study of the reliability of the developed DDS in the local network show that the use of the developed software allows to increase the level of reliability of detection by 5-12% compared to existing antivirus software, to reduce the level of errors of the first kind to 5% and improve the efficiency of its functioning when detected.

5. Conclusions

The principles of formation of architectures of distributed systems, in which the property of decentralization is synthesized, are analyzed. Also, the trends in the development of malware and trends in the development of anti-malware tools were analyzed.

As a result, distributed malware detection tools were proposed, in which the components would contain hardware and software. This provided an opportunity to improve the effectiveness of countering malware by 5%.

The proposed architecture of such tools can be used to build various types of malware detection tools in corporate networks, including honynet.

6. References

[1] O. Savenko, S. Lysenko, A. Kryschuk. Multi-agent based approach of botnet detection in computer systems. Communications in Computer and Information Science 291 (2012) 171-180 doi: 0.1007/978-3-642-31217-5_19

[2] T. Korkishko, A. Melnyk, Cryptographic processor architectures for DES algorithm, in: Proceedings of the 5th IEEE Africon Conference (Cat. No.99CH36342), Cape Town, South Africa, 1999, vol. 1, pp. 175-180. doi: 10.1109/AFRCON.1999.820788

[3] M. Bravo, G. Chockler, A. Gotsman, Making Byzantine consensus live. Distrib. Comput. 35, (2022) 503–532. doi:https://doi.org/10.1007/s00446-022-00432-y

[4] I. Abraham, G. Gueta, D. Malkhi, L.Alvisi, R. Kotla, J. Martin, Revisiting fast practical Byzantine fault tolerance (2017) arXiv:1712.01367

[5] Nunome, A., Hirata, H. Adaptive Parameter Tuning for Constructing Storage Tiers in an Autonomous Distributed Storage System. Int J Netw Distrib Comput 10 (2022) 1–10. doi: https://doi.org/10.1007/s44227-022-00004-3

[6] A. Nunome, H. Hirata, An adaptive tiering scheme for an autonomous distributed storage system. In: Proceedings of the 8th international virtual conference on applied computing and information technology (ACIT 2021), 2021, pp 62–68. ACM

[7] A. Nunome, H. Hirata, Performance evaluation of data migration policies for a distributed storage system with dynamic tiering. Int J Netw Distrib Comput (IJNDC) 8(1) (2019) 1–8

[8] K. Nozaki, T. Hochin, Evaluation of Identification Method of Corresponding Numerical Attributes in Heterogeneous Databases Based on Instances. Int J Netw Distrib Comput (2022). doi:https://doi.org/10.1007/s44227-022-00001-6

[9] K. Nozaki, T. Hochin, H. Nomiya, Identifcation of corresponding numerical attributes in heterogeneous databases based on instances. In: Proc. of 8th ACIS International Virtual Conference on Applied Computing & Information Technology (ACIT 2021), 2019

[10] HC. Huang, CH. Tsai, HC Lin, Development of 5G Cyber-Physical Production System. Int J Netw Distrib Comput (2022). doi:https://doi.org/10.1007

[11] F. Alrimawi, L. Pasquale, D. Mehta, N. Yoshioka, B. Nuseibeh Incidents are meant for learning, not repeating: sharing knowledge about security incidents in cyber-physical

systems. IEEE Trans Softw Eng 48(1) 120–134 doi: https://doi.org/10.1109/TSE.2020.2981310
[12] M. Azzam, et al. Forensic Readiness of Industrial Control Systems Under Stealthy Attacks. Computers & Security, 125, article no. 103010. doi:http://dx.doi.org/doi:10.1016/j.cose.2022.103010
[13] M. van Steen, A.S. Tanenbaum, A brief introduction to distributed systems. Computing 98, 967-1009 (2016). doi:https://doi.org/10.1007/s00607-016-0508-7J.
[14] S. Bernadette, T. Jared, A. Heather Harrington, N. Vidit, Geometric anomaly detection in data. Proceedings of the National Academy of Sciences (2020), 117(33) 19664-19669. doi: 10.1073/pnas.2001741117
[15] O. Pomorova, O. Savenko, S. Lysenko, A Kryshchuk, Multi-Agent Based Approach for Botnet Detection in a Corporate Area Network Using Fuzzy Logic, Communications in Computer and Information Science 370 (2013) 243-254
[16] S. Lysenko, O. Savenko, K. Bobrovnikova. DDoS Botnet Detection Technique Based on the Use of the Semi-Supervised Fuzzy c-Means Clustering. CEUR-WS 2104 (2018) 688-695
[17] O. Pomorova, O. Savenko, S. Lysenko, A. Nicheporuk. Metamorphic Viruses Detection Technique based on the the Modified Emulators. CEUR-WS 1614 (2016) 375-383
[18] O. Pomorova, O. Savenko, S. Lysenko, A. Kryshchuk, A.Nicheporuk, A Technique for detection of bots which are using polymorphic code, Communications in Computer and Information Science 431 (2014) 265-276
[19] Anson Steve. Applied Incident Response. John Wiley & Sons, Inc., 2020, 448 p.
[20] S. Bhunia, M. Tehranipoor, Hardware Security: A Hands-on Learning Approach. Morgan Kaufmann, 2019. — 501 p.
[21] S. Bhunia, S. Ray, S. Sur-Kolay, Fundamentals of IP and SoC Security: Design, Verification, and Debug. Springer, 2017, 316 p.
[22] D. Fitzpatrick, D. Bodeau, R. Graubart, R. McQuaid, C. Olin and J. Woodill, (DRAFT) Cyber Resiliency Evaluation Framework for Weapon Systems: Foundational Principles and Their Potential Effects on Adversaries, The MITRE Corporation, Bedford, MA, 2019.
[23] A. Mohanta, A. Saldanha, Malware Analysis Lab Setup. In: Malware Analysis and Detection Engineering. Apress, Berkeley, CA, 2020. doi: https://doi.org/10.1007/978-1-4842-6193-4_2
[24] A. Mohanta, A. Saldanha, Malware Components and Distribution. In: Malware Analysis and Detection Engineering. Apress, Berkeley, CA, 2010. doi:https://doi.org/10.1007/978-1-4842-6193-4_6
[25] E. Filiol, Viruses and Malware. In: Stavroulakis, P., Stamp, M. (eds) Handbook of Information and Communication Security. Springer, Berlin, Heidelberg, 2010. doi:https://doi.org/10.1007/978-3-642-04117-4_34
[26] M. Omar, Introduction to the Fascinating World of Malware Analysis. In: Defending Cyber Systems through Reverse Engineering of Criminal Malware. Springer Briefs in Computer Science. Springer, Cham. 2022. doi:https://doi.org/10.1007/978-3-031-11626-1_1
[27] Savenko O., Lysenko S., Nicheporuk A., Savenko B. Metamorphic Viruses' Detection Technique Based on the Equivalent Functional Block Search. CEUR-WS, ISSN: 1613–0073. 2017. Vol. 1844. – Pp. 555–569.
[28] J. Xue, Z. Wang and R. Feng, Malicious Network Software Detection Based on API Call, 2022 8th Annual International Conference on Network and Information Systems for Computers (ICNISC), Hangzhou, China, 2022, pp. 105-110, doi: 10.1109/ICNISC57059.2022.00032.
[29] W. Hu, J. Cheng, X. Chong, R. Zhang, B. Lin and A. Xia, A GAN-Based Anti-obfuscation Detection Method for Malicious Code, 2022 3rd International Conference on Pattern Recognition and Machine Learning (PRML), Chengdu, China, 2022, pp. 484-488, doi: 10.1109/PRML56267.2022.9882255.
B. Li, Research on the behavior detection technology of mobile software based on Big Data Mining, 2022 International Conference on Wearables, Sports and Lifestyle Management (WSLM), Kunming, China, 2022, pp. 12-16, doi: 10.1109/WSLM54683.2022.00008.