computing@computingonline.net
www.computingonline.net

Print ISSN 1727-6209
On-line ISSN 2312-5381
International Journal of Computing

THE METHODS OF ARTIFICIAL INTELLIGENCE FOR MALICIOUS APPLICATIONS DETECTION IN ANDROID OS

Sergei Bezobrazov [1)], Anatoly Sachenko [2)], Myroslav Komar [2)], Vladimir Rubanau [1)]

[1)] Brest State technical university, 224017, Moskovskaya 267, Brest, Belarus
bescase@gmail.com, vsrubanov@gmail.com
[2)] Ternopil National Economic University, 46020, 11 Lvivska st., Ternopil, Ukraine,
as@tneu.edu.ua, mko@tneu.edu.ua

Abstract: This paper presents and discusses a method for Android's applications classification with the purpose of malware detection. Based on the application of an Artificial Immune System and Artificial Neural Networks we propose the "antivirus" system especially for Android system that can detect and block undesirable and malicious applications. This system can be characterized by self-adaption and self-evolution and can detect even unknown and previously unseen malicious applications. The proposed system is the part of our team's big project named "Intelligent Cyber Defense System" that includes malware detection and classification module, intrusions detection and classification module, cloud security module and personal cryptography module. This paper contains the extended research that was presented during the IEEE 8th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS'2015) [1].

Keywords: Cyber Security, Artificial Intelligence, Android, Artificial Immune System, Artificial Neural Networks, Malware Detection.

1. INTRODUCTION

From year to year Android operating system increases its popularity. In June 2013 Google Inc. announced that it has over 1 billion active monthly Android users, up from 538 million this time last year. As of 2015 Android has the largest installed base of all general-purpose operating systems.

Android applications, that extend the functionality of devices, are written primarily in the Java programming language using the Android software development kit (SDK). The official store, Google Play Store, is the primary application store installed on Android devices that comply with Google's compatibility requirements and license the Google Mobile Services software. Google Play Store allows users to browse, download and update applications published by Google and third-party developers. As of July 2013 there are more than one million applications available for Android in Play Store and the application downloads grown to over 50 billion.

Android has a growing selection of third-party applications that can be acquired by users by downloading and installing the application's APK file, or by downloading them using an application store program that allows users to install, update, and remove applications from their devices.

Due to the open nature of Android, a number of third-party application marketplaces (such as Amazon Appstore, GetJar, SlideMe, F-Droid etc.) also exist for Android, either to provide a substitute for devices that are not allowed to ship with Google Play Store, provide applications that cannot be offered on Google Play Store due to policy violations, or for other reasons.

Such variety of Android applications and application stores makes the security problems very urgent.

Google Inc. developed the embedded security system that is based on implementation of a sandbox for the run of application and displaying all requires permissions for the installable application. Thus, for example, the weather application may need to enable save data or Internet connection, but should not need to read SMS messages or access the personal data. After reviewing the encountered permissions, the user can choose to accept or refuse the installation of an application. The implementation of the sandbox and the permissions system is lessens the impact of vulnerabilities but does not avoid them at all. The existing of different third-party stores do not increase the security level.

As a result, security threats on Android are reportedly growing exponentially. Thus, the security

company Trend Micro reported that the number of Android malware threats increased to a 25,000 samples in June 2015 [2]. It is interesting that in all of first quarter 2012, the number of malicious applications jumped by 5,000, while just one month in the second quarter 2012 discovered 10,000 samples. Another antivirus company, Kaspersky Lab, detected 35,000 malicious samples for the whole of 2012, while in first half of 2013 over 47,000 malicious applications were detected [3]. In the first half of 2014 alone, Kaspersky Lab experts detected 175,442 new unique malicious programs already.

In July 2015 the bug named 'Stagefright' was discovered in Android platform that allowed an attacker to perform arbitrary operations in the infected device through remote code execution and privilege escalation. Anyone can organize the attack on Android device using this bug by sending a specially crafted MMS messages.

The cybercriminals infect the mobile devices with the next aims: a) stealing money from the user's account; and b) stealing the information about the device owner, including all calls, correspondence, passwords to social network and e-pay accounts, etc.

It is well-known that current anti-viruses are characterized by several disadvantages. The most important of existing disadvantages are incapacity of signature-based methods to detect unknown malicious program and imperfection of used heuristic methods.

In our previous work [4]–[6] we developed the artificial immune system for malicious code detection for Microsoft Windows platforms and the artificial immune system for intrusion detection in computer networks. We integrated both system in the cyber defense system and showed that the developed systems have the ability of self-evolution and self-organizing and can detect not only already known but previously unseen computer viruses and network intrusions.

In this paper we present another new part of our 'Intelligent Cyber Defense System' that is directed to malicious applications detection in Android operating system. The proposed system is based on the integration of the Artificial Immune System and Artificial Neural Networks and aims to detect and block undesirable and malicious Android applications. The paper is organized as follows: Section 2 describes state-of-the-art in the area of the information security in Android platform. Section 3 contains the description of the proposed security system basic principles of work. Section 4 provides the experimental results. The final section concludes this paper. This article is an extended version of the paper that was presented during the IEEE 8th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS'2015) [1].

2. STATE-OF-THE-ART

In this section we describe the related research for classification of Android applications, their advantages and disadvantages. All methods for classification are divided into three groups that are close to the proposed research. There are immunity-based methods, permission-based methods and behaviour based methods.

2.1 IMMUNITY-BASED METHODS

M. Zhao et al. [7] proposed a theoretical framework to obtain and analyze smartphone application activity in Android environment. In collaboration with the Android user community, it will be capable of distinguishing between benign and malicious applications of the same name and version, detecting anomalous behavior of known applications. The authors have chosen artificial immunology algorithm to distinguish between benign applications and their corresponded malware version.

I. Vural et al. [8] proposed a technique based on artificial immune system to detect botnet spamming programs on Android phones. The main idea of the observed research is the ability of artificial immune systems to train only the positive examples. The authors have implemented a botnet detector based on artificial immune systems. The detector learns to classify valid SMSs from invalid. The botnet detector captures all outgoing SMSs, extracts the features from the message body and processes data to determine if the message is valid on not. When the detector encounters an SMS that it suspects to be invalid, it asks the user for confirmation that the message is valid. If the user confirms that, the detector sends an alert to the service provider. If the user indicates that the message is valid, then the detector learns to recognize the new pattern as a valid SMS.

2.2 PERMISSION-BASED METHODS

Felt et al. [9] performed studies to examine where they indicated that current Android permission warnings do not help most users make correct security decision. The authors developed the tool named 'Stowaway' [10], that applies static analysis on the collected sample applications, and then they map the permission with each operation. The aim of this is to detect over-privileged permissions in Android applications.

Chen et al. [11] proposed the system named 'Pegasus', in an attempt to detect malicious

applications that are characterized by the temporal order in which an application uses APIs and permissions. They constructed Permission Event Graph with static analysis and implemented models of the Android event-handling mechanism and APIs.

Enck et al. [12] constructed 9 permission rules called Kirin that classifies an application as potentially malicious if the application requests certain combinations of permissions that match the rules. The rules are defined by security requirement engineering.

2.3 BEHAVIOR-BASED METHODS

There is the set of similar behavior-based approaches for detection of malicious Android applications. Thus, I. Burguera et al. developed the behavior-based methodology in the form of an application, named 'Crowdroid' that collects systems calls of installed Android applications on the users' devices [13]. Then the collected systems calls are clustered using K-means algorithm into two categories: benign and malicious. The tests showed good results on the limited application set.

At the same time, T. Blasing et al. developed an Android emulator 'Android Application Sandbox' [14]. This tool logs the time of execution, the name of the system call and the IDs of the processes. Based on the collected information the tool makes the decision that the application is malicious or benign.

The main problem of the reviewed methods lies in their static structure. It means that for the keeping of high protection level it is necessary to update the system (different signature of malware, rules and behaviour signatures etc.) continuously. Despite the built-in security system the existing situation in the area of information security and cyber defense in Android OS is not characterized by strong protection and needs new strong solutions for detection of malicious applications. The application of methods of artificial intelligence allows making better the current situation and to create new methods for Android platform protection.

The next section presents our research in this area and describes the basic principles for creating the intelligent system for detection of malicious Android applications.

3. ARTIFICIAL IMMUNE SYSTEM AND ANDROID APPLICATION PACKAGE

In our previous works [4]-[6] we presented the 'Intelligent Cyber Defense System' that based on the integration of artificial immune networks and artificial immune system methods. Fig. 1 shows the generalized architecture of this system.

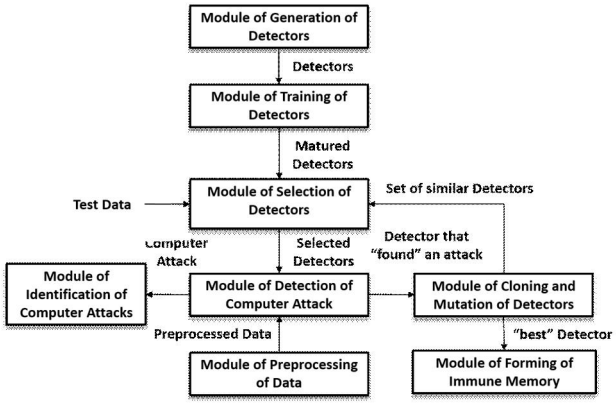

Fig. 1 – The generalized architecture of the 'Intelligent Cyber Defense System'

The proposed system consists of the set of "intelligent" immune detectors (each detector based on a neural network). Each detector has a time period, called lifetime, during that it is going through such stages as creation, training, selection, detection etc.

The *module of generation of detectors* produces the set of "immature" detectors. Then, during the *training stage,* the immune detectors are learned to correct classification of objects in the operating system environment and to detect a malicious code.

After being trained all immune detectors are going through the *selection stage* where their correctness is checked. If a detector classifies test benign objects as malicious, it is destroyed and replaced by a new detector. The module of selection allows decrease the false alarm rate and increase the defense level of the system.

Each new installed application is checked by the set of detectors that analyze it. If detectors classified the new application as benign, it can be installed on the device. If any detector classified an application as malicious, it is blocked.

When new malicious application was detected, the system extracts data from this application and adds it to the training data. Meanwhile, the detector that found a new malicious code is transformed into the immune memory detectors. The mechanism of updating the training data allows evolving the whole defense system and provides the ability to adapt to the new, unknown malicious applications. And, the mechanism of immune memory provides the high level of reaction on repeated attempts of known attacks.

We demonstrated that the system allowing detect not only known malicious objects (malware for Microsoft Windows platform in general) and network intrusions but also previously unseen and unknown computer threats. The main element of the mentioned system is an immune detector that represents a neural network. The developed

algorithms that are the basis of the system are universal and work excellent for malicious code detection as well as network intrusion detection (with insignificant corrections). Here is the choice of the structure of analyzed data play an important role. In the case of malicious code detection, a neural network immune detector analyze the source of files (for example, special areas of executable files) [4]. In the case of network intrusions detection, a detector analyzes a network traffic [5], [6]. Can we use the same approach for Android malicious applications detection and if yes, what data can we analyze? To answer this questions let's describe the structure of the typical Android application.

3.1 ANDROID APPLICATION PACKAGE

Android application package (APK) is the package file format used in Google's Android operating system. APK is used to distribute and install application software and is a type of archive file with .apk extension. Each APK archive usually contains such files as the manifest file, the certificate of the application, compiled code, etc. Fig. 2 represents the common structure of APK.

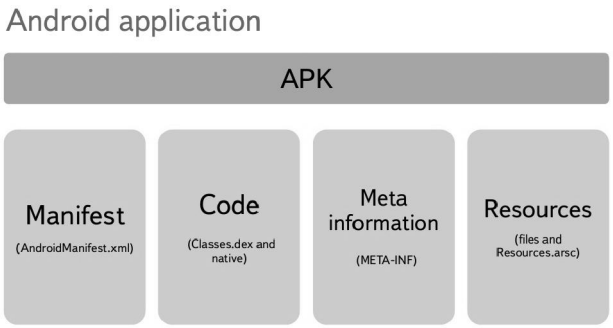

Fig. 2 – The structure of an Android application

From all encountered files of APK the file with the name AndroidManifest.xml is the most interesting and useful in the point of view of the current research and has the strong relation to the information security. Every Android application must have AndroidManifest.xml file (with precisely that name) in its root directory. The manifest file presents essential information about the application to the Android system. It provides such information as:

- describes the components of the application (the activities, services, broadcast receivers, and content providers that the application is composed of. It allows the Android system know what the components are and under what conditions they can be launched);
- determines which processes will host application components;
- declares which permissions the application must have in order to access protected parts of the API and interact with other applications;
- declares the permissions that others are required to have in order to interact with the application's components etc.

Without going into the detailed structure of the manifest file (it contains different elements such as permissions, instrumentations, configurations etc. [15]) says that this file contains very important information that can be used during analyzing and classification of Android applications. We are talking about the permissions.

A permission is a restriction limiting access to a part of the code or to data on the device. The limitation is imposed to protect critical data and code that could be misused to distort or damage the user experience. Each permission is identified by a unique label. Often the label indicates the action that's restricted [16]. For example, here are some permissions defined by Android:

- android.permission.INTERNET
- android.permission.ACCESS_FINE_LOCATION
- android.permission.VIBRATE
- android.permission.READ_PHONE_STATE
- android.permission.ACCESS_WIFI_STATE
- android.permission.READ_OWNER_DATA
- android.permission.DEVICE_POWER

A complete list of permissions is available on the Android Developers web site [16].

All permissions are divided into for protection levels (in accordance with the rules of the developers):

- 'normal' level contains lower-risk permissions which give requesting applications access to isolated application-level features, with minimal risk to other applications, the system, or the user. The system automatically grants this type of permission to a requesting application at installation, without asking for the user's explicit approval;

- 'dangerous' permission is a higher-risk permission that would give a requesting application access to private user data or control over the device that can negatively impact the user. Because this type of permission introduces potential risk, the system may not automatically grant it to the requesting application;

- 'signature' level contains permissions that the system grants only if the requesting application is signed with the same certificate as the application that declared the permission. If the certificates match, the system automatically grants the permission without notifying the user or asking for the user's explicit approval;

- finally, 'signatureOrSystem' permissions are the permissions that the system grants only to

applications that are in the Android system image or that are signed with the same certificate as the application that declared the permission.

The permissions are displayed to the user before application installation. Once the application is installed on the phone, there is no way to modify it. When the application is installed on the device, the installer determines whether or not to grant the requested permission by checking the authorities that signed the application's certificates. If the permission is granted, the application is able to use the protected features. If not, its attempts to access those features will simply fail without any notification to the user.

As it can be seen the permissions play an important role in the security in Android platform. In this way, using the permission that can be extracted from the Manifest file of each Android application we can construct the system that will allow us to analyze the behavior of the application and to classify them.

4. NEURONET IMMUNE DETECTORS FOR CLASSIFICATION OF ANDROID APPLICATIONS

In the previous section we provided the basic principles of our 'Intelligent Cyber Defense System' that is based on the integration of the methods of Artificial Neural Networks and Artificial Immune Systems. The main elements of this system are immune detectors that represent artificial neural network. The structure of the immune detector plays a key role in the classification and influences on the detection ability. In our opinion the neural network structure of immune detectors is preferable and it enables to construct more powerful detectors. We propose immune detectors that are based on the feed forward counter propagation neural network [17]. This neural network guarantees that it finds the correct weights during the learning process. Fig. 3 demonstrates the structure of the immune detector.

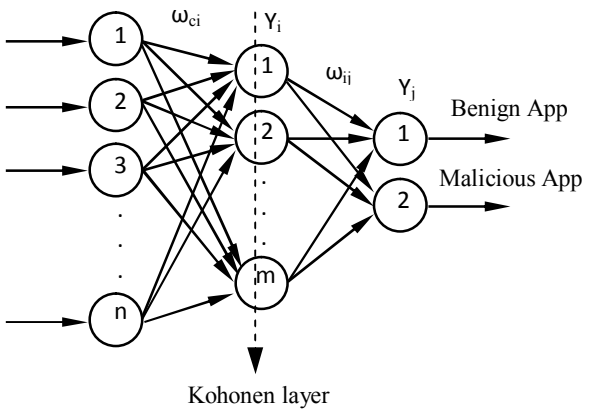

Fig. 3 – The structure of the neuronet immune detector

The proposed detector represents the neural network that consists of three layers of nodes. Input layer's nodes connect to each node in the hidden layer and receive data from outside (the set of the permission). The number of inputs nodes *n* equals to the size of the input vector (see Fig. 3).

The hidden layer consists of *m* Kohonen neurons and it represents a vector quantization layer [18], which gives the cluster label of the input pattern. The competitive learning rule (winner-takes-all) is used for training the hidden layer.

The output layer consists of linear units and carries out mapping of clusters into classes. It consists of two nodes: the activity of the first node indicates the legitimate object while the activity of the second node represents a malicious application.

During the training process immune detectors acquire the ability of classification of applications. The algorithm of the training of the immune detectors is described in [4]–[6] in details.

The proposed structure of immune detectors enables to use the small dataset for training and classify correctly real-world patterns after the training.

As a result, we receive the set of detectors that can classify any newly installed Android application.

5. EXPERIMENTAL RESULTS

In the Section 3 we described the structure of APK and demonstrated that the analyzing of data from a Manifest file allows classifying Android application. Each Android application contains the list of used permission. Accordingly, it is possible to construct a vector that will represent the information about which permissions are used and which permissions are not used in current application. As a result, each application can be represented as a binary vector The set of permission vectors forms the training set. During the training stage, each immune detector that is based on neural network is trained by the training set.

For the experiments we collected the benign and application from different sources. Then permissions were extracted and the permissions vectors for each application were formed. Fig. 4 shows the process of data extraction and Fig. 5 represents the example of the permissions' vector.

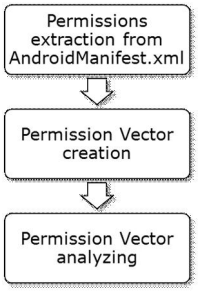

Fig. 4 – The process of data extraction

Then, each set of vectors (benign and malicious) was divided into two groups: training and testing set. Next, for each immune detector train data composed of benign and malicious vectors are created. After the learning, the set of immune detectors check the test data.

Application_1 (benin):

- android.permission.WAKE_LOCK
- android.permission.INTERNET
- android.permission.ACCESS_NETWORK_STATE
- android.permission.USE_CREDENTIALS
- android.permission.MANAGE_ACCOUNTS
- android.permission.READ_PHONE_STATE
- android.permission.ACCESS_WIFI_STATE
- android.permission.VIBRATE
- android.permission.GET_ACCOUNTS
- android.permission.WRITE_EXTERNAL_STORAGE

1	2	3	4	5	6	7	8	9	10	11	12	…	152
0	0	0	0	0	1	0	1	0	0	0	0	…	0

Application_2 (malicious):

- android.permission.RAISED_THREAD_PRIORITY
- android.permission.INTERNET
- android.permission.READ_PHONE_STATE
- android.permission.MOUNT_UNMOUNT_FILESYSTEMS
- android.permission.WRITE_EXTERNAL_STORAGE
- android.permission.WRITE_SECURE_SETTINGS
- android.permission.ACCESS_NETWORK_STATE
- android.permission.ACCESS_WIFI_STATE
- android.permission.CHANGE_NETWORK_STATE
- android.permission.RECEIVE_MMS
- android.permission.RECEIVE_WAP_PUSH
- android.permission.WRITE_SETTINGS
- android.permission.READ_SMS
- android.permission.SEND_SMS
- android.permission.RECEIVE_SMS

1	2	3	4	5	6	7	8	9	10	11	12	…	152
0	0	0	0	0	1	0	1	0	0	1	1	…	0

Fig. 5 – The vector of the permissions.

The results showed that immune detectors classify correctly not only applications from training set but also unknown applications.

5. CONCLUSION

The current research opens our big project directed on the creation of Intelligent Cyber Defense System based on the Artificial Immune System where immune detectors have a neural network structure. We already created the intelligent security system for detection of malicious code in Microsoft Windows platform and for detection of network intrusions. The system showed excellent results of unknown computer attacks detection. The developed apparatus can be successfully implemented for detection of malicious applications in Android operating system. We extracted the permissions from Manifest file and constructed the permission vector that can be used for training immune detectors fatherly and classify Android application. We received first results that confirmed our idea. Then, we are planning to develop the presented approach, improve the system for malicious application detection and integrate it in the already created Intelligent Cyber Defense System.

6. REFERENCES

[1] S. Bezobrazov, A. Sachenko, M. Komar, Vladimir Rubanau, "Artificial immune system for Android OS," in *Proceedings of the 8th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS'2015)*, Warsaw, Poland, 24-26 September 2015, pp. 403-407.

[2] Trend Micro: http://www.trendmicro.eu/ index.html, 2015.

[3] Kaspersky Lab: http://www.kaspersky.com, 2015.

[4] S. Bezobrazov, and V. Golovko, "Artificial immune systems of the neural network for the malicious code detection," in *Proceedings of the 6th International Conference on Neural Networks and Artificial Intelligence (ICNNAI'2010)*, Brest, Belarus, 2010.

[5] M. Komar, V. Golovko, A. Sachenko and S. Bezobrazov, "Intelligent system for detection of networking intrusion," in *Proceedings of the 6th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing System (IDAACS'2011)*, Prague, Czech Republic, 15-17 September 2011, pp. 374-377.

[6] V. Golovko, S. Bezobrazov, V. Melianchuk and M. Komar, "Evolution of immune detectors in intelligent security system for malware detection," in *Proceedings of the 6th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing System (IDAACS'2011)*, Prague, Czech Republic, 15-17 September 2011, pp. 722-726.

[7] M. Zhao et al, "A smartphone malware detection framework based on artificial immunology," *Journal of Networks*, Vol. 8, No. 2, pp. 469-476, February 2013.

[8] U. Vural, and H. Venter, "Detecting mobile spam botnets using artificial immune systems," *Advances in Digital Forensics VII*, pp. 183-192, August 1987 [Digests of the 9th Annual Conference on Magnetics, Japan, 1982, p. 301].

[9] A. Felt, E. Ha, S. Egelman, F. Haney, E. Chin, and D. Wagner, "Android permissions: user attention, comprehension, and behaviour," in *Proceedings of the Eight Symposium on Usable Privacy and Security (SOUPS'2012)*, 2012, p. 3.

[10] A. Felt, E. Chin, S. Hanna, D. Song, and D. Wagner, "Android permissions demistyfied," in *Proceedings of the ACM Conference on Computer and Communications Security*, Chicago, Illinois, USA, pp. 627-638, 2011.

[11] K. Z. Chen, N. M. Johnson, V. D'Silva, S. Dai, K. MacNamara, T. R. Magrino, E. Xue, J. Wu, M. Rinard, D. X. Song, "Contextual policy enforcement in Android applications with permission event graphs" in *Proceedings of the Network and Distributed System Security Symposium (NDSS'2013)*, 2013.

[12] W. Enck, M. Ongtang, and P. McDaniel, "On lightweight mobile phone application certification" in *Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS'09)*, 2009, pp. 235-245.

[13] I. Burguera, U. Zurutuza, and S. Nadjm-Tehrani, "Crowdroid: behavior-based malware detection system for Android," in *Proceedings the 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices*, Chicago, USA, 2011, pp. 15-26.

[14] T. Blasing, L. Batyuk, A. Schmidt, S. Camtepe, and S. Albaryak, "An Android Application Sandbox system for suspicious software detection," in *Proceedings the 5th International Conference on Malicious and Unwanted Software (MALWARE)*, Nancy, Lorraine, October 2010, pp. 55-62.

[15] App Manifest: http://developer.android.com/guide/topics/manifest/manifest-intro.html, 2015

[16] Manifest permission: http://developer.android.com/reference/android/Manifest.permission.html, 2015.

[17] S. Haykin, *Neural Networks: A Comprehensive Foundation*, Prentice Hall, 1999.

[18] T. Kohonen, *Self-organizing Maps. Springer Series in Information Sciences*, Springer, Berlin, Heidelberg, New York, Third Extended Edition, (30) 501, 2001.

***Sergei Bezobrazov**, PhD., associate professor of the Intelligent Information Technologies Department, Brest State Technical University. Scientific interests include artificial intelligence, machine learning, information security. He earned his B.Eng. Degree in Computer Science at Brest State Technical University in 2001 and his PhD Degree in Information Security at Belarus State University in 2006.*

His main areas of research interest include artificial intelligence, machine learning, information security. He has published over 40 scientific papers.

***Anatoly Sachenko** is Professor and Head of the Department of Information Computing Systems and Control and Research advisor of the Research Institute for Intelligent Computer Systems, Ternopil National Economic University. He earned his B.Eng. Degree in Electrical Engineering at L'viv Polytechnic Institute in1968 and his PhD Degree in Electrical Engineering at L'viv Physics and Mechanics Institute in 1978 and his Doctor of Technical Sciences Degree in Electrical and Computer Engineering at Leningrad Electrotechnic Institute in 1988. Since 1991 he has been Honored Inventor of Ukraine, since 1993 he has been IEEE Senior Member.*

His main Areas of Research Interest are Implementation of Artificial Neural Network, Distributed System and Network, Parallel Computing, Intelligent Controllers for Automated and Robotics Systems. He has published over 450 papers in areas above.

***Myroslav Komar** graduated Ternopil Academy of National Economy in 2001 with speciality "Information Systems in Management", 2002 – Master in Economical Cybernetics, 2013 – PhD degree.*

At the moment he is Associated Professor of the Department of Information Computing Systems and Control and Scientific Secretary of the Research Institute for Intelligent Computer Systems, Ternopil National Economic University.

Areas of scientific interests includes: Artificial Intelligent, Neural Networks, Immune System, Evolution of Systems, Cybersecurity.

***Vladimir Rubanau**, vice-rector of Brest State Technical University, PhD., Professor of the Higher Mathematics Department, PhD Degree at Belarus State University in 1986.*

His main areas of research interest include artificial intelligence, chaos theory.