Date of publication DEC-31, 2021, date of current version OCT-06, 2021.
www.computingonline.net / computing@computingonline.net

Print ISSN 1727-6209
Online ISSN 2312-5381
DOI 10.47839/ijc.20.4.2438

Architecture and Model of Neural Network Based Service for Choice of the Penetration Testing Tools

ARTEM TETSKYI[1], VYACHESLAV KHARCHENKO[1], DMYTRO UZUN[1], ARTEM NECHAUSOV[2]

[1]Department of Computer Systems, Networks and Cybersecurity, National Aerospace University KhAI,
Chkalova str., 17, 61070, Kharkov, Ukraine, {a.tetskiy, v.kharchenko, d.uzun}@csn.khai.edu, https://csn.khai.edu
[2]Department of Geo-information Technologies and Earth Monitoring, National Aerospace University KhAI,
Chkalova str., 17, 61070, Kharkov, Ukraine, a.nechausov@khai.edu, http://www.khai-gis.info

Corresponding author: Artem Tetskyi (e-mail: a.tetskiy@csn.khai.edu).

This research is supported by the project STARC (Methodology of SusTAinable Development and InfoRmation Technologies of Green Computing and Communication) funded by the Ministry of Education and Science of Ukraine.

ABSTRACT During penetration testing of web applications, different tools are actively used to relieve the tester from repeating monotonous operations. The difficulty of the choice is in the fact that there are tools with similar functionality, and it is hard to define which tool is best to choose for a particular case. In this paper, a solution of the problem with making a choice by creating a Web service that will use a neural network on the server side is proposed. The neural network is trained on data obtained from experts in the field of penetration testing. A trained neural network will be able to select tools in accordance with specified requirements. Examples of the operation of a neural network trained on a small sample of data are shown. The effect of the number of neural network learning epochs on the results of work is shown. An example of input data is given, in which the neural network could not select the tool due to insufficient data for training. The advantages of the method shown are the simplicity of implementation (the number of lines of code is used as a metric) and the possibility of using opinions about tools from various experts. The disadvantages include the search for data for training, the need for experimental selection of the parameters of the neural network and the possibility of situations where the neural network will not be able to select tool that meets the specified requirements.

KEYWORDS neural network; web service; cybersecurity; penetration testing; web applications; tools.

I. INTRODUCTION

PENETRATION testing is one of the ways to find Web application security problems [1]. While conducting such testing, tools intended to automate monotonic processes are actively used [2]. The problem is that for testing certain classes of Web application security problems, tools with similar functionality are used, and it is not known which tool is better to choose for a particular case. Usually, this is a problem of inexperienced testers who do not know the features of similar tools. Experienced testers usually have sets of tools that contain the most suitable tools for themselves. As a rule, one can choose the most suitable tool only empirically, that is, it is necessary to apply various means in certain conditions, analyze the results and draw appropriate conclusions.

Such comparative experiments are labor-intensive due to the large number of existing tools and a wide range of possible conditions of use. One of the methods of such comparison is the use of various tools on sites with previously known vulnerabilities [3]. The analysis of the several tools work results on such platform was given in [4]. It was found that the vulnerabilities of the Web application logic were not detected by any of the tools considered. This leads to the fact that the tools cannot do all the work for the tester and detect all the vulnerabilities.

A solution to the problem of choosing tools using a neural network is proposed in this paper. This method will allow the use of expert opinions about the feasibility of using various

tools as training data. The use of neural network models for solving such problems is analyzed in [5, 6]. Conclusions that can be made are that various types of neural network architectures can be used to solve classification problems. The most suitable architectures can be determined experimentally.

It is worth noting that artificial intelligence is already actively used in solving other problems related to the area of cybersecurity. In paper [7] a deep neural network based malware detection system was described. Authors of the paper [8] conducted an experiment to identify the unknown or zero-day malware, as a result of which malware was detected with high accuracy rate. Intrusion detection/prevention systems [9] may also use the neural network approach, as shown in [10].

Hence, there is a problem of choice of penetration tools for Web applications considering their features and requirements to cybersecurity. Besides, it should be taken into account that a set of tools is increased and tool characteristics are changed.

The goal of the paper is to describe the process of creating a neural network based Web service for choosing tools of penetration testing of Web applications. This paper is based on [11], in which prerequisites for creating a Web service for choosing penetration tools were described. Also, it discusses issues related to the implementation of neural network logic on the Web server side.

II. WEB SERVICE ARCHITECTURE

Fig. 1 shows the architecture of a Web service for choosing penetration testing tools. Software stack Linux + Apache + MySQL + PHP is used. Fast Artificial Neural Network (FANN) [12] is a library for creating neural networks. All technologies are free and often used to create Web applications.

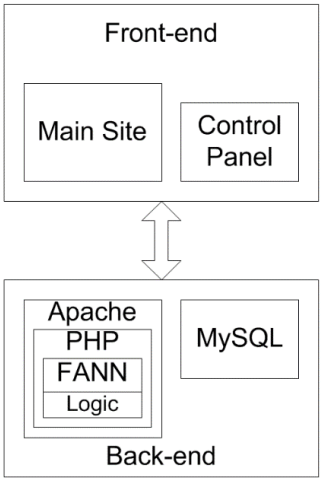

Figure 1. The structure of the neural network

The main part of the site is available to all users. By marking the criteria for the search, users form a vector of input data that is transmitted to the server. The neural network searches for tools that match the specified criteria, then a list of tools that, according to the neural network, correspond to the search query, is displayed to the user.

The control panel is a closed part of the site and is accessible only for the service administrator. The main functions of the control panel are as follows:

– Tool management. In the system there is a list of tools compiled on the basis of expert's opinions. With this feature, new tools can be added and existing ones can be edited or deleted.

– Criteria management. Each tool has a certain set of criteria. The total set of criteria is being edited by using this function.

– Management of opinions. One opinion consists of a set of criteria values of a tool. Using this function, administrator adds to the system the opinions about the tools received from the experts.

– Training neural network. After making changes to the list of opinions, the neural network should be retrained on actual data.

This functionality is due to the fact that the number of tools is growing, existing tools are updated, getting new functionality. The administrator must be in constant interaction with experts to provide a Web service with up-to-date information, adding it to the system as it becomes available.

All information about the tools is stored in a database. When training a neural network, data are being selected from the database, transformed into training data and, thus, the neural network is trained. The trained neural network is stored in a file; this is due to the functionality of the FANN library. If the retraining of the neural network is too resource-intensive, then it can be carried out in the hours of the least activity of the Web service so that the retraining does not affect the ability of the Web service to function.

III. BASELINE DATA FOR BUILDING A NEURAL NETWORK

The initial data for training the neural network, as already noted, are the opinions of experts in the field of testing security of Web applications about the tools that are used in penetration testing. These data are presented in the form of a table, where the rows are the criteria taken into account for tools selection process, and the columns are the tools themselves. An example of such data is shown in Table 1. The presented dataset is taken from [13], in which authors compared known scanners. Three scanners were numbered as T1, T2, and T3.

Table 1. Criteria table example

Criteria	T1	T2	T3
Server Side Java Script injection	1	0	0
Reflected Cross Site Scripting	1	1	1
Persistent Cross Site Scripting	1	1	1
DOM Cross Site Scripting	1	1	1
JSON Hijacking	0	1	0
Server-Side Includes Injection	0	1	1
Format String Attack	0	1	1
Code Injection	1	1	0
XML Injection	0	1	0

Forceful Browsing / Authentication Bypass	1	1	1
Privilege Escalation	0	1	0
Xml External Entity	1	1	0
Weak Session Identifier	0	1	1
Session Fixation	1	1	0
Cross Site Request Forgery	1	1	1

Traditionally, the structure of a neural network is defined, depending on the type of the problem being solved, and the set of training data can vary, which entails only a change in synaptic weights [14]. In this case, the structure of a neural network depends on a set of training data, since the number of input neurons is equal to the number of criteria, and the number of output neurons is equal to the number of tools that are available in the system.

IV. THE SEQUENCE OF CREATING A NEURAL NETWORK

Consider the process of creating a neural network using the training data from Table 1. It is proposed to use a structure with two hidden layers of neurons, which will allow identifying more complex dependencies than in the case of one hidden layer [15]. Such a decision may seem unreasonable, but it must be taken into account that in the practical application of the service, the number of tools can reach up to 100, the number of criteria is not known in advance. The possible number of tools in the system is justified by information about tools from the source [16], which currently contains a list of 43 tools that are often used to troubleshoot Web application security problems. The amount of training data presented in this work is not sufficient for the quality training of the neural network. It is impossible to determine how much data is needed to properly train a neural network; this is a separate topic for discussion and experimentation [17]. The data are shown to demonstrate the possibilities of using a neural network to solve the problem of choosing penetration testing tools.

There are no restrictions on the number of neurons in hidden layers; there are rules that can be followed when creating a neural network structure. One of these rules is the "geometric pyramid" rule [18], according to which the number of neurons in hidden layers is determined as follows:

$$r = \sqrt[3]{\frac{n}{m}}, \quad (1)$$

$$k_1 = mr^2, \quad (2)$$

$$k_2 = mr, \quad (3)$$

where n – the number of input neurons, m – the number of output neurons, k_1 – the number of neurons in first hidden layer, k_2 – the number of neurons in second hidden layer.

By applying the formulas (1)-(3) to calculate the number of neurons in hidden layers, the parameters of the neural network are the following:

– the number of input neurons n – 15 (corresponds to the number of tools criteria);

– the number of output neurons m – 3 (corresponds to the number of tools);

– the number of hidden layers – 2;

– the number of neurons in hidden layers k_1 and k_2 – 9 and 6 accordingly.

A graphical representation of the neural network structure is shown in Figure 2. In this case, the network is fully connected, i.e., sparseness coefficient is one.

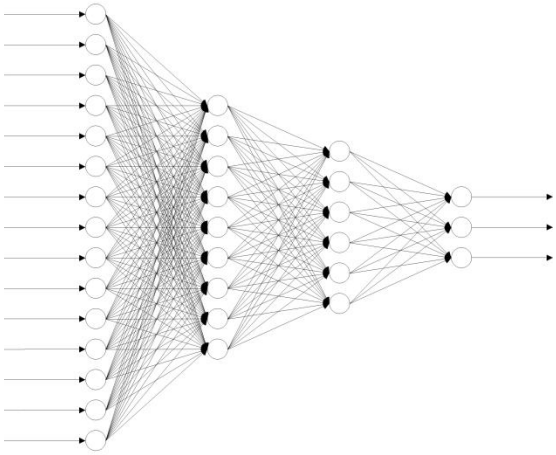

Figure 2. The structure of the neural network

One of the characteristics of the neural network is the activation function [19]. In this case, the sigmoid activation function is used, since it is often used in solving classification problems and has an output value in the range from zero to one.

V. TRAINING AND TESTING THE NEURAL NETWORK

The neural network is trained using the error propagation reverse method. In this paper, the Resilient Propagation (Rprop) algorithm is used, which is adaptive and does not require the denotation of training speed [20]. When learning, the number of learning epochs or the acceptable error value and the maximum number of learning epochs can be indicated if the value of the acceptable error was not reached in the learning process. As a rule, these parameters can only be determined experimentally.

The untrained network produces unstable results due to the initialization of synaptic weights with random values. If an increase in the number of epochs of learning does not lead to the required results, then, most likely, the reason of this is an insufficient amount of data for learning.

Let us transform values from Table 1 into training examples of a neural network. The result of the conversion is shown in Table 2. At the moment, one tool is characterized by one set of input data, since in each output data there is only one "1" in the corresponding position.

Table 2. Data for training

№	Input data	Output data
1	1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1	1, 0, 0
2	0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1	0, 1, 0
3	0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1	0, 0, 1

The neural network is tested using three examples, shown in Table 3.

Table 3. Test cases

№	Input data	Expected output data
1	1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1	1, 0, 0
2	1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0	1, 0, 0
3	0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0	1, 1, 1

In the first example, the input data matches one of the training examples, so the neural network should only select the first tool. In the second example, the neural network should select only the first tool, since the criteria corresponding to the first position of the input vector, which is present only in the first tool. The third example illustrates the search by the criterion that is present in all three tools.

To select the number of learning epochs, several training cycles and network testing will be conducted, increasing each time the number of learning epochs. The results obtained at 5000 epochs of learning are shown in Table 4. The higher the output value is, the higher the confidence of the neural network in the correct choice. As a rule, in practice, some limit value is used, for example, 0.8 or 0.9. If the output value is greater than the boundary value, then it is considered that the choice of a neural network can be trusted. For this particular case, the boundary value of 0.9 will be used.

Table 4. The results of work with 5000 epochs of study

Launch number	Sample number	Result
1	1	0.9938, 0.0000, 0.0055
	2	0.7782, 0.0001, 0.0873
	3	0.0519, 0.0054, 0.7701
2	1	0.9927, 0.0000, 0.0062
	2	0.7566, 0.0006, 0.0383
	3	0.0438, 0.0187, 0.5787
3	1	0.9922, 0.0062, 0.0000
	2	0.2827, 0.0179, 0.0254
	3	0.0046, 0.0475, 0.8524

In the first example, in all three launches the neural network made the right choice of tool (the first tool was chosen). In the second example, the highest output values also corresponded to the first tool, however, these values were not close to 1. In the third run of the second example, the minimum output value of 0.2827 was found, which indicates an insufficient number of learning epochs. In all launches of the third example, there is a tendency to choose the third tool, but the output values are not close to 1, i.e., do not exceed the boundary value.

The number of learning epochs was increased to 50000, the test results are shown in Table 5.

Table 5. The results of work with 50000 epochs of study

Launch number	Sample number	Result
1	1	0.9974, 0.0025, 0.0007
	2	0.9965, 0.0025, 0.0008
	3	0.8146, 0.0060, 0.0071
2	1	0.9985, 0.0013, 0.0009
	2	0.9585, 0.0060, 0.0032
	3	0.0523, 0.0244, 0.0690
3	1	0.9992, 0.0000, 0.0006
	2	0.9436, 0.0000, 0.0077
	3	0.0287, 0.0022, 0.6091

In all the tests of the first and second examples, it is clear that the result is stable and exceeds the boundary value. In the third example, the result remains unstable. Let us increase the number of learning epochs to 500000, the test results are shown in Table 6.

Table 6. The results of work with 500000 epochs of study

Launch number	Sample number	Result
1	1	0.9999, 0.0000, 0.0000
	2	0.9934, 0.0017, 0.0000
	3	0.0160, 0.1621, 0.0052
2	1	0.9999, 0.0001, 0.0000
	2	0.9181, 0.0000, 0.4310
	3	0.0003, 0.0002, 0.9991
3	1	0.9999, 0.0001, 0.0001
	2	0.9941, 0.0009, 0.0004
	3	0.0137, 0.0116, 0.0936

The first and second examples work stably, the third example continues to produce unstable results. Hence, the conclusion can be drawn that an increase of the number of learning epochs cannot solve the problem of the instability of work results. With the current training data set, the neural network cannot produce a stable result for the input data from the third example. It is necessary to expand the training examples number.

VI. FURTHER TRAINING OF THE NEURAL NETWORK

Let us assume that a new expert opinion has appeared about the second tool. The expert indicated the criteria inherent to this tool. A new training example is shown in Table 7.

Table 7. New training example

Input data	Output data
0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1	0, 1, 0

Let us train the network taking into account the new example and see how the result of the neural network operation will change in the third example. The number of

learning epochs is 500000, the test results are given in Table 8.

Due to the added training example, the neural network began to produce a stable result in the third test example. This can be explained by the fact that the second tool has 2 expert opinions about the availability of the required criteria, and the first and third tools have one expert opinion. In equal conditions it was impossible to determine which of the tools meets the specified requirements.

Table 8. The results of work with 500000 epochs of learning with the new training example

Launch number	Sample number	Result
1	1	0.9997, 0.0001, 0.0003
	2	0.9994, 0.0001, 0.0003
	3	0.0000, 0.9987, 0.0005
2	1	0.9997, 0.0001, 0.0003
	2	0.9997, 0.0001, 0.0003
	3	0.0000, 0.9973, 0.0001
3	1	0.9997, 0.0001, 0.0003
	2	0.9996, 0.0002, 0.0003
	3	0.0003, 0.9971, 0.0000

Examples given above show that increasing the quantity of learning epochs does not always lead to stable results. Overfitting of the neural network may also occur and it will entail the loss of network generalization possibility.

In practice, a large number of training examples are used to train a neural network. Cross-validation is used to evaluate the operation of a neural network on data that are not in the training set. During cross-validation, a lot of training data is divided into k identical blocks, at each iteration one of k blocks remains for model testing and k-1 blocks are used as training data [21]. The process is repeated k times, and each of the blocks is used once as a test set. The obtained data are combined to calculate an overall score.

VII. CONCLUSION

In this paper, the possibilities of using neural networks when choosing tools for penetration testing of Web applications are shown. The results of the neural network and their dependence on learning parameters are shown. The described mechanism is used on the Web server side and the FANN library is also used. The application of neural networks allows the use of expert opinions when choosing testing tools.

The advantage of using neural networks is the simplicity of implementation in comparison with deterministic algorithms; the number of lines of code is used as metrics parameter. The opinions of various experts are taken for training, thus avoiding the subjectivity of opinions. There may be a situation where some opinions will be contradictory. In such circumstances, the neural network may not select a tool that meets the search requirements. The disadvantages include the need for experimental selection of neural network parameters. It is also difficult to find data for training due to the high requirements for experts who provide training data for the neural network.

Further research may be related to the accumulation and processing of information from experts as the set of analyzed tools expands. In addition, cases should be investigated when a neural network makes a decision that this set of tools does not allow them to be selected according to specified criteria.

References

[1] M. Vieira, N. Antunes and H. Madeira, "Using web security scanners to detect vulnerabilities in web services," in *Proceedings of the 2009 IEEE/IFIP International Conference on Dependable Systems & Networks*, Lisbon, Portugal, June 29 - July 2, 2009, pp. 566-571. https://doi.org/10.1109/DSN.2009.5270294.

[2] N. Awang and A. Manaf, "Detecting vulnerabilities in web applications using automated black box and manual penetration testing," in *Proceedings of the International Conference on Advances in Security of Information and Communication Networks SecNet'2013,* Cairo, Egypt, September 3-5, 2013, pp. 230-239. https://doi.org/10.1007/978-3-642-40597-6_20.

[3] F. R. Muñoz, I. I. S. Cortes and L. J. G. Villalba, "Enlargement of vulnerable web applications for testing," *The Journal of Supercomputing*, vol. 74, issue 12, pp. 6598-6617, 2018. https://doi.org/10.1007/s11227-017-1981-2.

[4] A. Doupé, M. Cova and G. Vigna, "Why Johnny can't pentest: An analysis of black-box web vulnerability scanners," in *Proceedings of the International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment DIMVA'2010,* Bonn, Germany, July 8-9, 2010, pp. 111-131. https://doi.org/10.1007/978-3-642-14215-4_7.

[5] M. C. Nicoletti, J. R. Bertini Jr., D. Elizondo, L. Franco and J. M. Jerez, "Constructive neural network algorithms for feedforward architectures suitable for classification tasks," in: L. Franco, D. A. Elizondo, J. M. Jerez (Eds.), *Constructive Neural Networks*, Berlin, Heidelberg, 2010, pp. 1-23. https://doi.org/10.1007/978-3-642-04512-7_1.

[6] R. Sadeghian and M. R. Sadeghian, "A decision support system based on artificial neural network and fuzzy analytic network process for selection of machine tools in a flexible manufacturing system," *International Journal of Advanced Manufacturing Technology*, vol. 82, issue 9-12, pp. 1795-1803, 2016. https://doi.org/10.1007/s00170-015-7440-4.

[7] J. Saxe and K. Berlin, "Deep neural network based malware detection using two dimensional binary program features," in *Proceedings of the 2015 10th International Conference on Malicious and Unwanted Software (MALWARE)*, Fajardo, Puerto Rico, October 20-22, 2015, pp. 11-20. https://doi.org/10.1109/MALWARE.2015.7413680.

[8] M. Alazab, S. Venkatraman, S. Watters and M. Alazab, "Zero-day malware detection based on supervised learning algorithms of API call signatures," in *Proceedings of the Ninth Australasian Data Mining Conference*, vol. 121, Ballarat, Australia, December 1-2, 2011, pp. 171-182.

[9] A. S. Ashoor and S. Gore, "Difference between intrusion detection system (IDS) and intrusion prevention system (IPS)," in *Proceedings of the International Conference on Network Security and Applications,* Chennai, India, July 15-17, 2011, pp. 497-501. https://doi.org/10.1007/978-3-642-22540-6_48.

[10] S. S. Roy, A. Mallik, R. Gulati, M. S. Obaidat and P. V. Krishna, "A deep learning based artificial neural network approach for intrusion detection," in *Proceedings of the International Conference on Mathematics and Computing*, Haldia, India, January 17-21, 2017, pp. 44-53. https://doi.org/10.1007/978-981-10-4642-1_5.

[11] A. Tetskyi, V. Kharchenko and D. Uzun, "Neural networks based choice of tools for penetration testing of web applications," in *Proceedings of the 2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies (DESSERT'2018),* Kyiv, Ukraine, May 24-27, 2018, pp. 402-405. https://doi.org/10.1109/DESSERT.2018.8409167.

[12] S. Nissen and E. Nemerson, *Fast Artificial Neural Network Library (FANN),* [Online]. Available at: http://leenissen.dk/fann/html/files/fann-h.html

[13] M. Mirjalili, A. Nowroozi and M. Alidoosti, "A survey on web penetration test," *Advances in Computer Science: An International Journal*, Los Alamitos, CA, vol. 3, issue 6, no. 12, pp. 107-121, 2014.
[14] J. E. Dayhoff and J. M. DeLeo, "Artificial neural networks: opening the black box," *Cancer: Interdisciplinary International Journal of the American Cancer Society*, vol. 91, no. S8, pp. 1615-1635, 2001. https://doi.org/10.1002/1097-0142(20010415)91:8+<1615::AID-CNCR1175>3.0.CO;2-L.
[15] C. Y. Chen, J. R. C. Hsu and C. W. Chen, "Fuzzy logic derivation of neural network models with time delays in subsystems," *International Journal on Artificial Intelligence Tools*, vol. 14, no. 6, pp. 967-974, 2005. https://doi.org/10.1142/S021821300500248X.
[16] *Kali Linux Tools Listing*, 2019, [Online]. Available at: https://tools.kali.org/tools-listing
[17] H. Park and S. Baek, "An empirical validation of a neural network model for software effort estimation," *Expert Systems with Applications: An International Journal*, vol. 35, issue 3, pp. 929-937, 2008. https://doi.org/10.1016/j.eswa.2007.08.001.
[18] T. Masters, *Practical Neural Network Recipes in C++*, Morgan Kaufmann, 1993, 493 p. https://doi.org/10.1016/B978-0-08-051433-8.50017-3.
[19] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, Sardinia, Italy, May 13-15, 2010, pp. 249-256.
[20] C. Igel and M. Hüsken, *Improving the Rprop Learning Algorithm*, in: H. Bothe, R. Rojas (Eds.), *Proceedings of the Second International ICSC Symposium on Neural Computation (NC 2000)*, vol. 2000, ICSC Academic Press, 2000, pp. 115-121.
[21] R. Setiono, "Feedforward neural network construction using cross validation," *Neural Computation*, vol. 13, no. 12, pp. 2865-2877, 2001. https://doi.org/10.1162/089976601317098565.

Artem Tetskyi, graduated National Aerospace University *"KhAI" with a* master's degree in Computer Systems and Networks. Now he works as an Assistant Lecturer of Computer Systems, Networks and Cybersecurity Department of National Aerospace University. Research interests: information technologies, Web development, cybersecurity, fuzzy logic.

Prof. Vyacheslav Kharchenko is a head of Computer Systems, Networks and Cybersecurity department at National Aerospace University *"KhAI". Research interests:* dependable computing, safety and security critical software and FPGA based systems modeling; software reliability assessment and prediction using SRGMs and big data analysis; green information technologies; fault and intrusion tolerant embedded systems and IoT.

Dr. Dmytro Uzun, Associate Professor of Computer Systems, Networks and Cybersecurity Department of National *Aerospace University "KhAI". His* experience includes information systems maintenance, system administration, development for new information system, hardware and software assessment and selection, technical expertise, research in the field of disk less technologies and open source software technologies.

Ph.D., Artem Nechausov, received his *master's degree with honours,* qualification level of geo-informatics engineer in 2013 and PhD degree in Information Technologies from National Aerospace University "KhAI" in 2016. Now he works as an Associate Professor of the Department of Geo-information Technologies and Earth Monitoring at the Faculty of Rocket and Space Technology in National Aerospace University "KhAI*". *Ukraine. Research interests:* Information technologies, GIS Analysis, geomarketing, gravity models, Web development, fuzzy logic, aerospace monitoring.