Automated Software Vulnerability Testing Using Deep Learning Methods

Alexandr Kuznetsov
V. N. Karazin Kharkiv National University
Kharkiv, Ukraine
kuznetsov@karazin.ua

Yehor Yeromin
V. N. Karazin Kharkiv National University
Kharkiv, Ukraine
suvenick2@gmail.com

Oleksiy Shapoval
V. N. Karazin Kharkiv National University
Kharkiv, Ukraine
alex.shapoval@protonmail.com

Kyrylo Chernov
V. N. Karazin Kharkiv National University
Kharkiv, Ukraine
kirillfilippsky@gmail.com

Mariia Popova
V. N. Karazin Kharkiv National University
Kharkiv, Ukraine
mariia.popova26@gmail.com

Kostyantyn Serdukov
Kyiv National University of Trade and Economics
Kyiv, Ukraine
kfuduefin@gmail.com

***Abstract*—The paper provides a state of current technologies, which are used for automated vulnerability testing in software. Features of fuzzing technology (which is based on making many inputs with different mutated data) are also studied. The essence of the algorithm is the selection of input data, which are more likely to cause a failure or incorrect behavior of the software product. Deep learning algorithms are used to reduce the computational complexity of the testing process. Using a simple fuzzer and Deep Reinforcement Learning algorithm shows that the number of mutations required to detect vulnerabilities is reduced by 30%.**

Keywords*—*Fuzzing, Testing, Reinforcement Learning, Q-Learning, software security.

I. Introduction

The development of modern computer technology leads to the emergence of new high-quality information services and their implementation in all spheres of human activity [1-5]. The development of the IT-industry has led to the construction of global computer networks, extensive data warehouses, automated control systems, including critical infrastructures, Smart Grid and much more [6-8].

In the era of the Internet and the global information technologies, information security becomes more important for critical infrastructures [9, 10]. A problems comprehensive solution related to information security is associated with solving various problems in cryptography [11-14], computing optimization, technical and physical security, as well as many others [10, 15-17].

This article is aimed on the problem of automatic software vulnerabilities scanning [18-22]. As practice shows, software is the most vulnerable part of modern IT infrastructure. Errors or configuration errors and undeclared operations can have disastrous consequences. Development and research of methods and tools for automatic software vulnerabilities scanning is an extremely important and urgent task.

II. Intellectual Fuzzing Algorithm

Fuzzing is a method for testing software vulnerabilities, which is performed by conducting several tests using modified input data [21]. Retesting is performed with a random mutation, and usually the testing time is far from optimal. This article addresses the problem of intellectual fuzzing and attempts to find a solution [22]. The main purpose is to develop a technology that can be guided and that will make decisions grounded on the experience gained during the testing. The solution to this question lies in machine learning and reinforcement learning based on the deep Q-learning algorithm [23]. It uses the maximum possible remuneration, which is determined in the development process by analyzing the initial data of the program and the available rewards. This allows to apply optimal input data mutations. Thus, agent gets an opportunity to learn to formulate an optimal action policy for obtaining maximum reward. In this article, we describe an algorithm and a computer model of in-depth training, and a study of the effectiveness of automatic vulnerabilities testing compared to the random mutations test. During testing, the "black box" method is used. It means that the available information represents only results of the program's work and the input data that it needs to perform [21].

As a combination of training fuzzing and reinforcement, was created a system that can change the rules for choosing a particular mutation. It sends the changed data to the input and, depending on the program response, generates a reward in order to lean on its own experience and select the optimal mutations for a particular case with further testing. Thus, the number of mutations that do not contribute to the testing process is significantly decreased. This makes the testing process faster. Schematic diagram of the developed system is shown in Fig. 1.

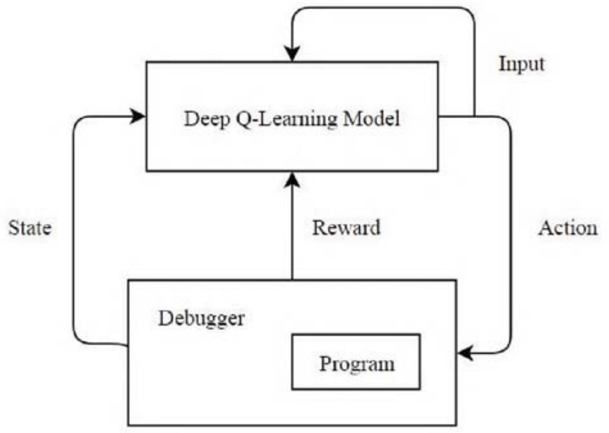


Fig. 1. Simplified diagram of the intellectual fuzzing algorithm

The testing process begins with the determination of the initial non-mutant input data. Its format depends on fuzzing type. Packages are sent to the input channel of the program, and the program's response is obtained by using special debugging software. From the received data (this can be the program execution time, code coverage, code completion, etc.) the system state is formed. It will be pre-processed and submitted to the contribution of the Deep Q-learning model. The system then decides what action to take next.

At the same time, depending on the chosen action and the received state of the program, the system generates a reward for the algorithm. Using this reward, the algorithm understands the task and determines the optimal behavior for its implementation. In addition, the algorithm remembers which actions brought him the maximum reward (error detection, program crash, etc.), and in the next testing cycles it decides what actions should be taken based on its already acquired experience. To calculate the next step, the previous input data mutates according to the selected action. Program input receives new mutated data. This procedure is repeated until the algorithm reaches its goal. Developed model uses Markov decision-making process – deep Q-learning [25].

III. Reinforcement Learning

This section provides necessary data about the algorithm of reinforcement training. Reinforcement learning is a computational approach to understanding and automating targeted learning and decision making. It stands out among other machine learning algorithms by the fact that agent learns directly by interacting with the environment without referring to examples [25]. This algorithm is primarily aimed to solve problems that arise during interaction with the environment to achieve long-term action. It uses formal structure of Markov decision-making process [25], defining the interaction between agent and environment in terms of states, actions and rewards. These features include understanding of causes and effects, as well as presence of clear goals. The notion of value and function of value is the main features of reinforcement training methods.

As noted earlier, the interaction between agent and environment can be described using Markov decision-making process

$$M = (S, A, P),$$

where S – a set of system's states, A – action's set, P – transition probabilities set.

For each state-action pair $(s,a) \in S \times A$, P is a set of probabilities $P(s' \vee s, a)$, where s' corresponds to the next system's state. Agent considers possible system states for the selected action, where each transition has its own reward $r(s,a)$, and studies the optimal behavior for maximizing the reward.

During the learning process, the main goal is to maximize the finite amount of rewards:

$$R = \sum_{t=0}^{\infty} \gamma^t r_{t+1},$$

where $\gamma \in (0,1)$ – discount rate, which determines the remuneration priority over time. Action a_t at state s_t is determined by the policy of action $a_{t\pi}(\vee s_t)$. Policy π attaches considered possible states to action, which in turn determines agent's behavior. Expected cumulative reward for the policy-maker π is defined as:

$$Q^{\pi}(s,a) = E\left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} \vee s_0 = s, a_0 = a\right].$$

The problem of finding the optimal value $Q_{\pi}(s,a)$ can be reduced to the procedure of function approximation. To achieve this $Q_{\pi}(s,a)$ needs to be updated after each iteration of receiving the award [26]. It is defined as

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha,$$

where α – learning rate.

The entire procedure can be stated in the following sequence: agent gets the status s_t, takes action

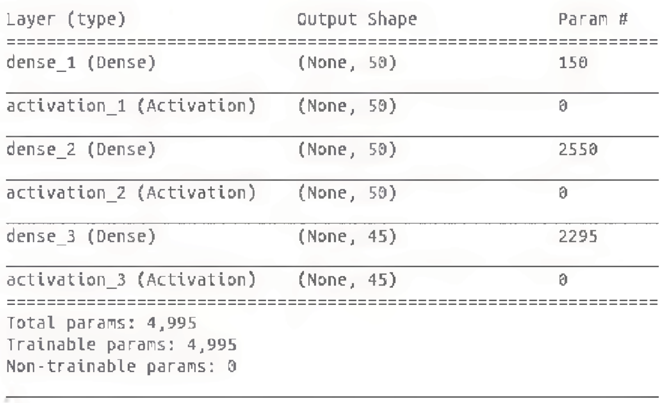
$$a_t = \arg\max(Q(s_t, a))$$

which defines the reward r_t, and causes the system to go to the state $s_t + 1$. After receiving a reward r_t and state $s_t + 1$, agent determines the best possible effect

$$a_t + 1 = \arg\max(Q(s, a)).$$

IV. Simulation Results

To conduct an experiment simple fuzzer was used. Its main function is to implement input data selection for automated software vulnerability search. Software launch process was simulated and special logic tests were developed. The program could return an error code when certain data is received to compare fuzzing with and without artificial intelligence. Possible states of the system are presented in the form of data generated after the completion of the program. This data is transmitted by neural network, which consists of one input layer, two hidden layers with 50 neurons each and activation functions (Rectified Linear Unit): $f(x) = \max(0, x)$. The output of the neural network has 45 elements, representing the number of possible mutations. Complete scheme of neural network for function approximation $Q(s_t, a_t)$ is shown in Fig. 2.



```
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 50)                150
_________________________________________________________________
activation_1 (Activation)    (None, 50)                0
_________________________________________________________________
dense_2 (Dense)              (None, 50)                2550
_________________________________________________________________
activation_2 (Activation)    (None, 50)                0
_________________________________________________________________
dense_3 (Dense)              (None, 45)                2295
_________________________________________________________________
activation_3 (Activation)    (None, 45)                0
=================================================================
Total params: 4,995
Trainable params: 4,995
Non-trainable params: 0
_________________________________________________________________
```

Fig. 2. Neural network scheme

Input mutations were selected based on the standard list: increasing and decreasing line length, integer insertion, adding special characters (for example, "%s", which may also cause errors). In sum, 45 functions were created and placed in a dictionary for further use.

The system receives an award if execution time was greater than the previous or if there were errors occurred during testing. When an error occurs, the algorithm finishes its work. While forming this type of award, there was a problem when algorithm already found one error and started calling it repeatedly to get maximum reward. To avoid this problem, two constants must be set: $\gamma \in (0,1)$ – discount rate and $\varepsilon \in (0,1)$ – greed factor. The first constant was already being mentioned before. The second one determines how the algorithm is capable in terms of discovering new solutions. Random action will be chosen with probability ε, and the most profitable action will be selected with probability $1 - \varepsilon$. The discount rate is set at 0.9, and the greed factor is equal to 0.5. The latter parameter decreases 0.99 times after each epoch. A special tests for identifying an error was developed. The tests were divided into a training set and a validation set. Each test is a function that returns an error code, or the program hangs with certain input data and represent a long chain of if-else blocks. Errors were arranged in a special way that would require 20-30 specific mutations in a row. This was done so that the difficulty of finding such an error was high. But not so much that it was impossible to find the error for random action choosing. Tests were done for both numeric and string types of data. The main idea of the tests was to check whether our algorithm can find errors of an unknown type, based on the actions that led to errors in previous tests. Also, some of the actions that were defined had no consequences. This was done in order to see whether the algorithm will be able to identify them as non-carrier rewards and not use them in validation (where the greed factor is set to 0). In the training set, there were 90 different tests, in the validation set - 10.

We developed all tests and network architecture in Python language with using frameworks such as tensorflow and Keras for building our model and training phase. We wrote three modules: environment where we specify our model and Q-learning algorithm which shown in Fig.3, actions where we specified the list of actions and tests module where all the tests are located.

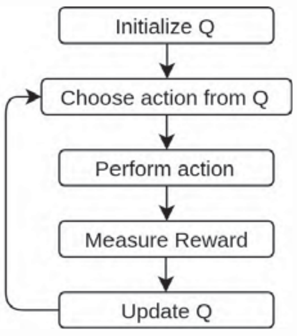


Fig. 3. Q-learning algorithm

During training, the algorithm searches for optimal Q-values for each of the 45 prescribed actions based on the award received. Network training is performed using Adam optimization algorithm [26] and learning rate set to 0.002 after the hyperparameter optimization stage, which are selected by trial and error (layers count, learning rate, neurons count and the number of epochs.

Pseudocode for training phase is shown in Fig. 3.

The algorithm trained 20 epochs, after which it showed results in the form of an average number of steps for finding the error 964 for the train set and 1634 for the validation set. We find these results good enough to make a final test and conduct a comparative statistical experiment with previously unseen tests. For this, we made an algorithm that randomly selects actions from the same list of actions. The hypothesis of the experiment is Q-learning fuzzing works faster than randomized one. The hypothesis is a scientific assumption that made to explain any phenomenon and requires testing on theoretical basis in order to become a reliable scientific theory [27].

```
//set total reward to 0, reset the environment state,
set current state to environment state, set max mutations to 5000
total_reward = 0
environment.reset()
state = environment.state
t_max = 5000
// do while error not found or in range t_max
for t in range(t_max):
    // choose action with greedy-policy
    action = get_action(state, epsilon=epsilon)
    // perform action, get next state and reward
    next_state, reward, done = env.step(action)
    // check if training perform Adam optimizer step
    if train: train_step()
    // add current reward to total reward
    total_reward += reward
    // go into next state
    state = next_state
    // check if error was found end.
    if done: break
```

Fig. 4. Pseudocode for training phase

The following sequence of actions was necessary to test the hypotheses:

1. Make calculations of certain statistics, the distribution of which is known.
2. Find the P-value for the calculated results.
3. Make appropriate conclusions depending on the significance criterion and P-value.

The Student's t-test was used [28] to test the hypothesis. Student's t-test – the general name for the class of methods for statistical criteria testing. It is based on comparison with the Student’s distribution. The most common application of the criteria is related to checking the equality of mean values in two samples [28]. To use this criteria, some conditions must be met: the initial data must have normal distribution and dispersion must be equal.

The first group contains the results of testing using developed algorithm, while the second one presents the results of random mutations. Testing was performed in the following sequence: generate 15 experiments, and record the amount of mutations necessary to find an error at the end of each. Under the experiment, we assume the search for errors in our last test, which our model has not yet seen. This is repeated 15 times to increase statistical confidence and reduce variance.

The results of the experiments are shown in Table 1.

TABLE I. EXPERIMENTAL RESULTS

№	Deep Q-learning model	Random selection of mutations
1	3671	3686
2	1191	1897
3	1879	3164
4	1640	3233
5	1966	10446
6	1585	5358
7	1135	1134
8	4877	752
9	2465	2157
10	3266	3684
11	1895	2026
12	2093	2993
13	1150	295
14	1181	3381
15	1153	358

The results of Student's t-test calculation: $t = -12.40$.

The number of degrees of freedom:

$$v = 2n - 2 = 2 \times 15 - 2 = 28 .$$

Considering significance criteria $\alpha = 0.01$ and $P-value = 2.763$, While $t < P-value$, the hypothesis is valid.

The result is statistically significant for a given criterion, if the probability of accidental occurrence of the same or extreme result is less than the given level (0.01) under the condition of loyalty of the null hypothesis.

The testing time of developed algorithm is better than the time of random mutations testing considering the fact the algorithm did not learn before. On average, developed algorithm finds an error after 2076 mutations, whereas random testing needs 2832 mutations. Represented algorithm finds error 30% faster.

This result proves the direction of research was chosen right. Particularly, it shows that the main problem of fuzzing (large amount of mutations) can be solved using artificial intelligence methods [29-34]. Actually, if input data is changed directionally depending on previous results, it can speed up mutation process and receive results (which are finding the vulnerability in certain software product) after less amount of mutations. Thus, the results obtained can be used in various methods and mechanisms for ensuring information security [35-44].

V. CONCLUSIONS

Due to conducted research, one of promising automated software testing methods in critically important systems was analyzed. Fuzzing bases on multiple input of different (mutated) data to find parameters, which will cause failure or incorrect functioning of software. Repeated testing is usually carried using randomized mutations and the time of testing is very high in the most cases. This article researches the problem of intellectual fuzzing – the technology, which uses previous testing experience to make choices, related to mutation, and reduce testing time.

Deep Reinforcement Learning algorithm was used to implement intellectual fuzzing. With the use of simple fuzzing app, it becomes possible to prove that testing time decreases by 30%. This result was received because fuzzer used previous experience to adjust mutations.

REFERENCES

[1] M. J. Bitner, V. A. Zeithaml, D. D. Gremler, P. P. Maglio, C. A. Kieliszewski, J. C. Spohrer, "Technology's Impact on the Gaps Model of Service Quality" in Handbook of Service Science - Service Science: Research and Innovation in the Service Economy, US, Boston, MA: Springer, pp. 197-217, 2010.

[2] J.vom Brocke, B. Thurnher, D. Winkler, "Improving IT-Service Business Processes by Integrating Mobility: The Role of User Participation-Results from Five Case Studies", *International Conference on eLearning e-Business Learning e-Business Enterprise Information Systems and e-Government (EEE 2008)*, 2008.

[3] J. Budzik, K. Hammond, "Watson: Anticipating and Contextualizing Information Needs", *Proc. of the 62 Annual Meeting of the American Society for Inform. Science*, 1999.

[4] "World Manufacturing Production 2012", *United Nations Industrial Development Organization*, 2013.

[5] "Structural Change in the World Economy", *United Nations Industrial Development Organization*, 2009.

[6] IEEE Vision for Smart Grid Communications: 2030 and Beyond," in IEEE Vision for Smart Grid Communications: 2030 and Beyond, vol., no., pp.1-390, 31 May 2013.

[7] A. Albarakati, B. Moussa, M. Debbabi, A. Youssef, B. L. Agba and M. Kassouf, "OpenStack-Based Evaluation Framework for Smart Grid Cyber Security," *2018 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, Aalborg, 2018, pp. 1-6.

[8] M. E. Hariri, T. Youssef, H. F. Habib and O. Mohammed, "A Network-in-the-Loop Framework to Analyze Cyber and Physical Information Flow in Smart Grids," *2018 IEEE Innovative Smart Grid Technologies - Asia (ISGT Asia)*, Singapore, 2018, pp. 646-651.

[9] S. S. Sokolov, N. B. Glebov, E. N. Antonova and A. P. Nyrkov, "The Safety Assessment of Critical Infrastructure Control System," *2018 IEEE International Conference "Quality Management, Transport and Information Security, Information Technologies" (IT&QM&IS)*, St. Petersburg, 2018, pp. 154-157.

[10] Krasnobayev V., Kuznetsov A., Koshman S., Moroz S. (2019) Improved Method of Determining the Alternative Set of Numbers in Residue Number System. *In: Chertov O., Mylovanov T., Kondratenko Y., Kacprzyk J., Kreinovich V., Stefanuk V. (eds) Recent Developments in Data Science and Intelligent Analysis of Information. ICDSIAI 2018. Advances in Intelligent Systems and Computing*, vol 836. Springer, Cham, pp. 319-328, 05 August 2018. DOI: 10.1007/978-3-319-97885-7_31

[11] "Cybersecurity for Smart Grid Systems". (25 June 2018) [On-line]. Internet: https://www.nist.gov/programs-projects/cybersecurity-smart-grid-systems.

[12] Andrushkevych A., Gorbenko Y., Kuznetsov O., Oliynykov R., Rodinko M. A (2019) "A Prospective Lightweight Block Cipher for Green IT Engineering". *In: Kharchenko V., Kondratenko Y., Kacprzyk J. (eds) Green IT Engineering: Social, Business and Industrial Applications. Studies in Systems, Decision and Control*, vol

171. Springer, Cham, pp. 95-112. DOI: 10.1007/978-3-030-00253-4_5

[13] N. Ferguson and B. Schneier. *Practical Cryptography*. John Wiley & Sons, 2003, 432 p.

[14] Kuznetsov O., Potii O., Perepelitsyn A., Ivanenko D., Poluyanenko N. (2019) "Lightweight Stream Ciphers for Green IT Engineering". *In: Kharchenko V., Kondratenko Y., Kacprzyk J. (eds) Green IT Engineering: Social, Business and Industrial Applications. Studies in Systems, Decision and Control*, vol 171. Springer, Cham, pp. 113-137. DOI: 10.1007/978-3-030-00253-4_6

[15] H. Yinghua, M. Yanchun and Z. Dongfang, "The Multi-join Query Optimization for Smart Grid Data," *2015 8th International Conference on Intelligent Computation Technology and Automation (ICICTA)*, Nanchang, 2015, pp. 1004-1007.

[16] Zu Du. Intelligence Computation and Evolutionary Computation. Springer-Verlag Berlin Heidelberg, 2013, 1114 p.

[17] Chong Tong, Yuan Gao, Minguang Tong, Jingru Li, Qin Wang and Kai Chen, "Application of lightning detection in Smart Grid dynamic protection," *2012 International Conference on Lightning Protection (ICLP)*, Vienna, 2012, pp. 1-13.

[18] S. Y. Choi and H. Y. Lee, "Toward Automated Scanning for Code Injection Vulnerabilities in HTML5-Based Mobile Apps," *2016 International Conference on Software Security and Assurance (ICSSA)*, St. Polten, 2016, pp. 24-24.

[19] H. Peng, Y. Shoshitaishvili and M. Payer, "T-Fuzz: Fuzzing by Program Transformation," *2018 IEEE Symposium on Security and Privacy (SP)*, San Francisco, CA, 2018, pp. 697-710.

[20] J. Zhao and L. Pang, "Automated Fuzz Generators for High-Coverage Tests Based on Program Branch Predications," *2018 IEEE Third International Conference on Data Science in Cyberspace (DSC)*, Guangzhou, 2018, pp. 514-520.

[21] M. Sutton, A. Greene, and P. Amini. "Fuzzing: Brute Force Vulnerability Discovery", 1st ed. Boston, MA, USA: Addison-Wesley Professional, 2007. [On-line]. Internet: http://bxi.es

[22] Konstantin Böttinger, Patrice Godefroid, Rishabh Singh. "Deep Reinforcement Fuzzing", Submitted on 14 Jan 2018, 2018. [On-line]. Internet: https://arxiv.org/abs/1801.04589.

[23] Yuxi Li. "Deep Reinforcement Learning: An Overview", Submitted on 25 Jan 2017 (v1), last revised 26 Nov 2018 (this version, v6), 2018. [On-line]. Internet: https://arxiv.org/abs/1701.07274.

[24] "AlphaGo Games". [On-line]. Internet: https://deepmind.com/research/alphago/match-archive/alphago-games-english/.

[25] R. S. Sutton and A. G. Barto. "Reinforcement learning: An introduction". MIT press Cambridge, 1998, 427 p. [On-line]. Internet: http://incompleteideas.net/book/bookdraft2017nov5.pdf.

[26] Diederik P. Kingma, Jimmy Ba. "Adam: A Method for Stochastic Optimization". Submitted on 22 Dec 2014 (v1), last revised 30 Jan 2017 (this version, v9). [On-line]. Internet: https://arxiv.org/abs/1412.6980.

[27] V.S. Pugachev, "Probability Theory and Mathematical Statistics for Engineers". Pergamon, 1984, 468 pp, ISBN: 978-0-08-029148-2.

[28] J. Goodier, "The Concise Encyclopedia of Statistics". Heidelberg: Springer 2008. ix+616 pp., ISBN: 978-0-387-31742-7.

[29] Alexandr Kuznetsov, Oleksiy Shapoval, Kyrylo Chernov, Yehor Yeromin, Mariia Popova, Olga Syniavska. Automated software vulnerability testing using in-depth training methods. In Proceedings of the Second International Workshop on Computer Modeling and Intelligent Systems (CMIS-2019), Zaporizhzhia, Ukraine, April 15-19, 2019., pp. 227–240. 2019.

[30] A.A. Kuznetsov, Smirnov, A.A., D.A. Danilenko, A. Berezovsky. "The statistical analysis of a network traffic for the intrusion detection and prevention systems." *Telecommunications and Radio Engineering*, Volume 74, 2015 Issue 1, pp. 61-78. DOI: 10.1615/TelecomRadEng.v74.i1.60

[31] V. Panchenko, A. Zamula, S. Kavun and I. Mikheev, "Intelligent management of the enterprise personnel security system," *2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, Kyiv, Ukraine, 2018, pp. 469-474.

[32] O. Oksiiuk and V. Chaikovska, "Development of authentication process when accessing cloud services," *2017 4th International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PIC S&T)*, Kharkov, 2017, pp. 604-607. doi: 10.1109/INFOCOMMST.2017.8246473

[33] O. Oksiiuk, L. Tereikovska and I. Tereikovskiy, "Determination of expected output signals of the neural network model intended for image recognition," *2017 4th International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PIC S&T)*, Kharkov, 2017, pp. 596-599. doi: 10.1109/INFOCOMMST.2017.8246471

[34] V. Krasnobayev, S. Koshman, A. Yanko and A. Martynenko, "Method of Error Control of the Information Presented in the Modular Number System," *2018 International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PIC S&T)*, Kharkiv, Ukraine, 2018, pp. 39-42. doi: 10.1109/INFOCOMMST.2018.8632049

[35] Krasnobayev V. A. Method for realization of transformations in public-key cryptography. *Telecommunications and Radio Engineering. - Volume 66, 2007 Issue 17,* pp. 1559-1572.

[36] Runovski, K., & Schmeisser, H. -. (2004). On the convergence of fourier means and interpolation means. Journal of Computational Analysis and Applications, 6(3), 211-227.

[37] Gnatyuk, V. A. (2001). Mechanism of laser damage of transparent semiconductors. Physica B: Condensed Matter, 308-310, 935-938. doi:10.1016/S0921-4526(01)00865-1

[38] Tkach, B. P., & Urmancheva, L. B. (2009). Numerical-analytic method for finding solutions of systems with distributed parameters and integral condition. Nonlinear Oscillations, 12(1), 113-122. doi:10.1007/s11072-009-0064-6

[39] O. Oksiiuk, L. Tereikovska and I. Tereikovskiy, "Adaptation of the neural network model to the identification of the cyberattacks type "denial of service"," *2018 14th International Conference on Advanced Trends in Radioelecrtronics, Telecommunications and Computer Engineering (TCSET)*, Lviv-Slavske, 2018, pp. 502-505. doi: 10.1109/TCSET.2018.8336251

[40] O. Oleksandr, L. Myrutenko, Y. Shestak and G. Viktoria, "Formation of the Method of Branched its Power Distribution by Activities and Specifics of Work," *2018 International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PIC S&T)*, Kharkiv, Ukraine, 2018, pp. 95-98. doi: 10.1109/INFOCOMMST.2018.8632025

[41] S. Toliupa, O. Oksiuk and N. Lukova-Chuiko, "Choice of reasonable variant of sygnal and code constructions for multirays radio channels," *2015 Second International Scientific-Practical Conference Problems of Infocommunications Science and Technology (PIC S&T)*, Kharkiv, 2015, pp. 269-271. doi: 10.1109/INFOCOMMST.2015.7357333

[42] O. J. Onyigwang, Y. Shestak and A. Oksiuk, "Information protection of data processing center against cyber attacks," *2016 IEEE First International Conference on Data Stream Mining & Processing (DSMP)*, Lviv, 2016, pp. 397-400. doi: 10.1109/DSMP.2016.7583586

[43] J. O. Ogbu and A. Oksiuk, "Information protection of data processing center against cyber attacks," *2016 Third International Scientific-Practical Conference Problems of Infocommunications Science and Technology (PIC S&T)*, Kharkiv, 2016, pp. 132-134. doi: 10.1109/INFOCOMMST.2016.7905358

[44] Bychenkov, V., Koretskyi, A., Oksiiuk, O., Vialkova, V. Assessment of capabilities of military groupings (forces) based on the functional group "engage". Eastern-European Journal of Enterprise Technologies