UDC 004.9 **doi:** https://doi.org/10.20998/2522-9052.2021.3.16

Serhii Semenov[1], Cao Weilin [2], Zhang Liqiang [2], Serhii Bulba[1]

[1] National Technical University "Kharkiv Polytechnic Institute", Kharkiv, Ukraine
[2] Neijiang Normal University, Neijiang, China

AUTOMATED PENETRATION TESTING METHOD USING DEEP MACHINE LEARNING TECHNOLOGY

Abstract. The article developed a method for automated penetration testing using deep machine learning technology. The main purpose of the development is to improve the security of computer systems. To achieve this goal, the analysis of existing penetration testing methods was carried out and their main disadvantages were identified. They are mainly related to the subjectivity of assessments in the case of manual testing. In cases of automated testing, most authors confirm the fact that there is no unified effective solution for the procedures used. This contradiction is resolved using intelligent methods of analysis. It is proposed that the developed method be based on deep reinforcement learning technology. To achieve the main goal, a study was carried out of the Shadov system's ability to collect factual data for designing attack trees, as well as the Mulval platform for generating attack trees. A method for forming a matrix of cyber intrusions using the Mulval tool has been developed. The Deep Q - Lerning Network method has been improved for analyzing the cyber intrusion matrix and finding the optimal attack trajectory. In the study, according to the deep reinforcement learning method, the reward scores assigned to each node, according to the CVSS rating, were used. This made it possible to shrink the attack trees and identify an attack with a greater likelihood of occurring. A comparative study of the automated penetration testing method was carried out. The practical possibility of using the developed method to improve the security of a computer system has been revealed.

Keywords: machine learning; software security; automated penetration testing.

Introduction

Using computer systems in almost all in all areas in social life and the increase in the number of information security incidents have updated the problem of data and software protection. One of the ways to improve cybersecurity is through the use of penetration testing methods and tools. This applies both to the areas of real production and practical services, and to the area of software development.

Until recently, penetration testing was done manually. At the same time, first of all, based on their own experience and erudition, the testers needed to analyze the computer system to detect vulnerabilities. Only then can you enter the system and compromise the software. This is a rather laborious task, on the one hand, it requires a large amount of tester knowledge, and on the other hand, it has many risks of a subjective nature. Therefore, more and more organizations have recently been using penetration attack planner options based on automated targeting system models. So, for example, the company Core Security using this idea since 2010 in its tool Core IMPACT uses the attack planner MetricFF [1]. In addition, Core Security began the practice of implementing certain ethical cyberattacks in accordance with the known exploits of software vulnerabilities [2]. However, no uniform solution has been obtained for penetration testing procedures.

One of the ways to solve this contradiction is to use attack tree methods. These methods are based on the provisions of the theory of graphs by Bruce Schneier [3], which introduced the informal concept of attack trees to systematize and categorize various attack scenarios on computer systems.

By analyzing the attack tree, penetration testers can visualize and understand the interaction between attack patterns. For example, the paper [4] proposes a graph-based approach for modeling and detecting attacks. At the same time, transitions from state to state along the extended attack tree are characterized by the attributes of the TTL lifetime and the degree of PC confidence. The first attribute reflects the temporal dependencies between the stages of the attack and helps to reduce the number of false positives of the cyber intrusion detection system. The second one characterizes the probability of achieving the goal with the achieved sub-goals. It should be noted that such a prototype of ethical cyber intrusions illustrates attack scenarios, but does not reduce the number of false positives. In [5] A formal definition of information attack trees is given [6] using disjunctive Petri nets. In this model, places correspond to attacks, and transitions often express logical dependencies, thereby simulating the actions of intruders, which extends this model compared to Schneier's attack trees.

The studies carried out have shown that simultaneously with the study of models and methods for synthesizing attack trees, it is advisable to analyze them. This is a prerequisite for optimizing and reducing their visual complexity. The result of the analysis is determined by the purpose of the attack graph. For penetration testing examples, such analysis should find the most likely attack scenarios for the attacker.

Among the many methods for analyzing attack graphs, the following can be distinguished. The authors of paper [7] define the shortest paths to goals using Dijkstra's algorithm. The set of least cost paths is also computed by the Naor-Brutlag algorithm. It is shown the problem of determining numerous effective protection measures (by cost) is NP-hard. The following interactive technique is proposed: the administrator changes the network model in the configuration file in accordance with the adopted security measures, and then recalculates the shortest paths to see if these changes have increased the security level of the network or not. However, the authors of the article created a

realistic-sized attack graph based on 10 or 20 patterns, and did not resolve all the problems associated with matching patterns with the attacker's configuration and profile. In addition, the presence of attack patterns and a configuration file can also be another vulnerability. If they fall into the wrong hands, they can be valuable tools for an attacker.

And in paper [8] the author conducts an analysis aimed at finding the minimum critical set of attacks - a set of attacks of the lowest power, the removal of which from the violator's arsenal will make it impossible for him to reach the goal. It is shown that the problem of finding such a set is NP-complete, and a "greedy" heuristic algorithm for solving it of complexity O (*mn*) is proposed, where m is the number of states and edges, n is the number of attacks. At the same time, as the authors themselves note, the presented development does not cover the entire variety of methods for analyzing and optimizing graphs, which presents an opportunity for researchers to further improve.

In [9], to solve the problem of analyzing an attack tree, the authors used the machine learning method. At the same time, *Q* was taken as a basis - training for finding the trajectory of attacks. However, the insignificance of the action space and the sample space reduced the practical value of this development. According to several authors [10], deep reinforcement learning is a useful advancement in attack tree analysis for penetration testing. For example, in paper [11], the task of analyzing and optimizing the attack graph using this intelligent technology is performed. At the same time, the width of the spectrum of possible vulnerabilities, as well as the shortcomings of penetration testing automation methods to ensure the security of computer systems

The aim of the work is an automated penetration testing method using deep machine learning technology. For this, it is necessary to solve a number of particular scientific problems:

1) Explore the factual data collection capabilities of the Shadov system for the design of attack trees, as well as the Mulval platform for generating attack trees.

2) Develop a method for forming a matrix of cyber intrusions using the Mulval tool.

3) Improve the Deep *Q* - Lerning Network method for analyzing cyber intrusion matrices and finding the optimal attack trajectory.

4) Conduct a comparative study of the automated penetration testing method.

The general structure of the method can be represented as a set of methods and tools in Fig. 1. As shown in Figure 1, the structure of the method implies the presence of three components: a set of technologies and means of forming an attack matrix; deep reinforcement learning for processing attack matrix data; tools, platforms and penetration testing tools.

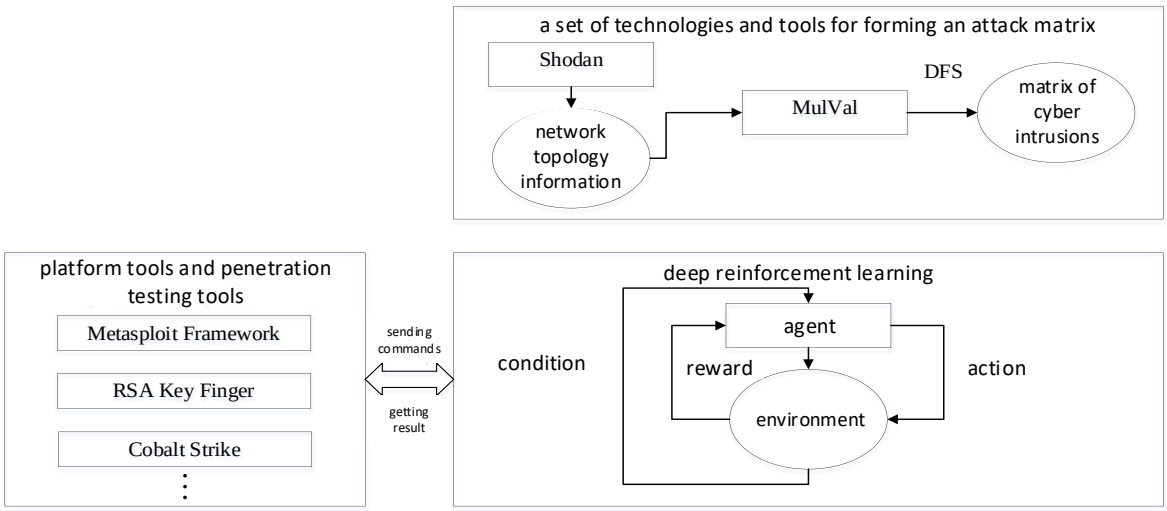

Fig. 1. General structure of the automated penetration testing method

1. Investigation of the capabilities of the Shadov and Mulval platform

It is known that the Shodan system is designed to work with shadow Internet channels. The system is not looking for web resources with content, but physical devices connected to the Internet. These can be printers, webcams, routers, GPS navigators, and even commercial maintenance systems. The basic principle of Shodan is to send requests to all publicly available IP addresses and log their responses. Algorithm for scanning this system:

1) generating a random IP address;

2) selection of a random port number from the list of ports available in Shodan;

3) checking the selected IP-address (port) and getting a banner;

4) repeating step 1.

Thus, the system scans the entire address space at random to ensure even coverage of the Internet and prevent data from shifting at any time. Shodan also supports searching for information about software vulnerabilities [12].

The system also allows you to select several criteria for searching and filtering data to monitor the current state of the IS. The main Shodan filters are: City / country (filtering devices located within the specified city / country, for example city: minsk); Port (output devices with a given open port, for example port: 443);

OS (filtering devices that run on a given operating system, for example os: linux); Geo (exact indication of the coordinates of the device location (longitude, latitude), for example geo: 42.9693,74.1224); Net (search for devices from a given range of IP addresses, for example net: 216.0.0.0/16) [12].

Mulval is a widely used network security analysis tool that uses a vulnerability scanner to find network vulnerabilities and then generates attack graphs for security analysis [6]. The block diagram of the framework is shown in Fig. 2.

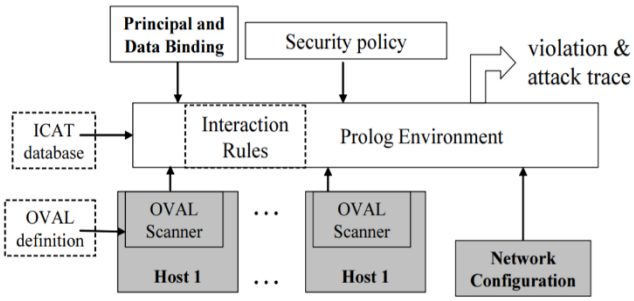

Fig. 2. Block diagram of Mulval framework

2. Development of a method for forming a matrix using the Mulval tool

2.1. A method of forming an attack tree. Having received an informal concept in 1999 in the work of Bruce Schneiter, attack trees, as a methodology for describing security threats, are now widespread. The attack tree is characterized as follows: the root of the tree corresponds to the intruder's goal, and the vertices correspond to the intruder's actions to achieve the goal. Actions are combined using logical AND and OR. Vertices and edges can be assigned different numbers, all values that characterize the probability of success, complexity, cost of the offender's actions. In Fig. 3 shows a graph and a textual representation of each of the possible types of nodes, AND and OR.

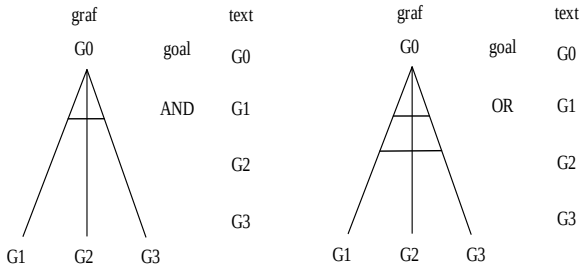

Fig. 3. Graph and text representation of each of the possible node types, AND and OR

Considering the above facts, it should be noted that the first stage - modeling of the attack tree, is very important for the formation of training data. This data, in turn, is extremely important for use in deep learning algorithms. For the formation of input data in the work, it is proposed to perform the following relevant stages.

1) Gathering network information to simulate a network environment using Shodan.

2) Generation of an attack tree corresponding to this network environment using the Mulval platform.

3) Data preprocessing (formation of a matrix of attacks) to adapt to the requirements of deep learning algorithms.

The first step is to generate the necessary practical query in the Shodan system, such as a query for information on real Web servers. An example of data collected through Shodan (excluding IP addresses) is shown in Fig. 4. Based on the data obtained, a profile file is created with the necessary information about the network node. An example of a web server profile is shown in Table 1.

"info": "(CentOS)",
"ip_str": 192.168.1.1,
"isp": HiNet,
"os": null,
"port": 443,
"product": "Apache httpd",
"transport": "tcp",
"version": "2.2.15",
"vulns": {
"CVE-2010-1452",
…
}

Fig. 4. Sample data collected via Shodan

Table 1 – **Example of a web server profile**

Product	Port	Protocol	Vulnerability	OS
Apache	80	https	CVE-2010-1452	CentOS
Nginx	8080	https	CVE-2011-0419	Ubuntu
mt-daapd DAAP	3689	tcp	CVE-2017-9617	FreeBSD

In addition to the dataset presented in Table 1, a vulnerability file is also generated. The data includes:

- identifier number of vulnerabilities CVE, Microsoft, etc.
- component of the basic scoring type
- exploitablityScore fitness score component

Table 2 provides an example of a vulnerability dataset. At the second step of generating the attack tree, it is proposed to use the well-known Mulval platform. The Mulval model is based on logic programming and is described using the DataLog language, which is a dialect and a subset of the Prolog language.

This language is implemented by the IDE XSB. When generating on the basis of Mulval, the following types of objects-vertices of the attack graph are distinguished - inference vertices (inference rule) and fact vertices (rules are predicates that determine the values and type of action prototype). The semantic meaning of the scenario stage lies in the logical following "Conjunct → Fact", which allows you to determine when the selected predicate has the value "true". A conjunct is a logical conclusion.

An example of generating an attack tree using the Mulval tool is shown in Fig. 5.

2.2. Algorithm for transforming an attack tree into a cyber intrusion matrix. The next component of the automated penetration testing method being developed is an algorithm for transforming an attack tree into a cyber intrusion matrix. For the research, the algorithm for transforming the attack tree into the corresponding matrix, developed by the authors [10], was chosen as a prototype. The study of the method carried out on the basis of [10] made it possible to make a choice about the limitations of its use for penetration testing. This is largely due to authors's neglect of inputs such as "file access", "command execution", and focusing only on vulnerabilities.

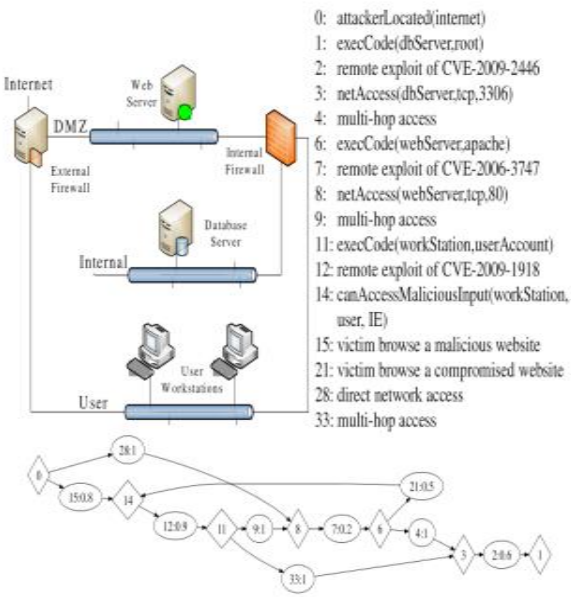

Fig. 5. An example of an attack graph scenario developed using the Mulval tool

Below is a description of the improved algorithm for transforming attack trees into a cyber intrusion matrix. At the first step of the algorithm, all nodes in the attack tree are matched into a matrix form.

At the second step, the complex compares quantitative scoring of security vulnerabilities to the baseline scoring of the corresponding nodes. The baseline score is calculated in accordance with the CVSS (Common Vulnerability Scoring System) structure. According to [13], the scores are calculated using special formulas based on metrics and characterize the ease of deployment of an exploit and its impact on a computer system. As an improvement, it is proposed to superimpose on the base CVSS score some predefined exploitability score, (for example, "file access") in accordance with the expression:

$$ovsc = bsc \times \frac{expsc}{5}, \qquad (1)$$

where $ovsc$ – overall security vulnerability scoring, bsc – baseline security vulnerability scoring, $expsc$ – serviceability scoring.

At this step, instead of embedding the cyber intrusion matrix directly into the deep machine learning algorithm, it is proposed to simplify this matrix using a modified DFS algorithm - Tarjan's algorithm [14]. This algorithm is primarily one of the options for depth-first search. In this case, the vertices are visited from roots to leaves, and the end of their processing is performed on the way back.

This simplification of the complete matrix allows you to select only those nodes that can be used to achieve the target of the attack.

At the end of the execution of the Taryan algorithm, the third step is to form a simplified cyber intrusion matrix, in which the following are written:

- score values for the start node in the first column;

- the values of the total scores characterizing the intermediate stages in the intermediate columns;

- the score values for the end node in the last column.

This data will be used as input to evaluate the reward in the deep machine learning algorithm.

2.3. Deep Reinforcement Learning to Determine the Optimal Attack Trajectory. In the developed method of automatic penetration testing, the method of deep reinforcement learning is used to determine the optimal attack trajectory through continuous learning.

The input data for the method is formed on the basis of a simplified cyber intrusion matrix, and the soft max function, a logistic function for the multidimensional case, will serve as the prototype of the activation function:

$$\nu(z)_i = e^{z_i} \Big/ \sum_{k=1}^{K} e^{z_k}, \qquad (2)$$

where K – is the number of classes.

During training, a deep reinforcement learning (DQL) model agent acts as an ethical hacker, while the targeted system is represented as a simplified cyber intrusion matrix. The agent moves from node to node in the developed matrix until it reaches the desired result - the "victim's server". The bonus factors used in the DQL modeling process correspond to the values of the vulnerability assessment based on the data of the Common Vulnerability Scoring System (CVSS) (1). In CVSS, a baseline assessment interprets the severity of the negative impacts of particular types of vulnerabilities, while an exploitability assessment captures the potential for exploitation of that particular vulnerability. To obtain the most appropriate result from a practical point of view, it is advisable to carry out mathematical weighting of these estimates by multiplying by the appropriate coefficients. The values of the coefficients are determined empirically, taking into account the opinion of experts in the field of cybersecurity.

2.4. Experimental studies of automated penetration testing method. To assess the performance and effectiveness of the developed method of automated software penetration testing, a series of experiments was carried out. In this case, the targeted system was presented in the form of a topological structure of a computer network in Fig. 6.

From the presented diagram, it can be seen that the targeted system is a collection of three different services, including a web server and mail server on one subnet, as well as a file server on another subnet. The client on the first subnet is controlled by Windows 2000, and the client on the second subnet is controlled by Internet Explorer. In the presented diagram, a workstation with Acrobat software is also connected to the same subnet with the file server.

As a means of protecting a computer network from cyber intrusions, a firewall is used in the scheme.

The rules for connecting to this tool are as follows.

1. The agent is located in the external network and has the ability to access the web server through the HTTP protocol and the corresponding HTTP port.

2. There is a dual connection between the web server and workstations on the network.

3. Web server and file server are connected via NFS protocol.

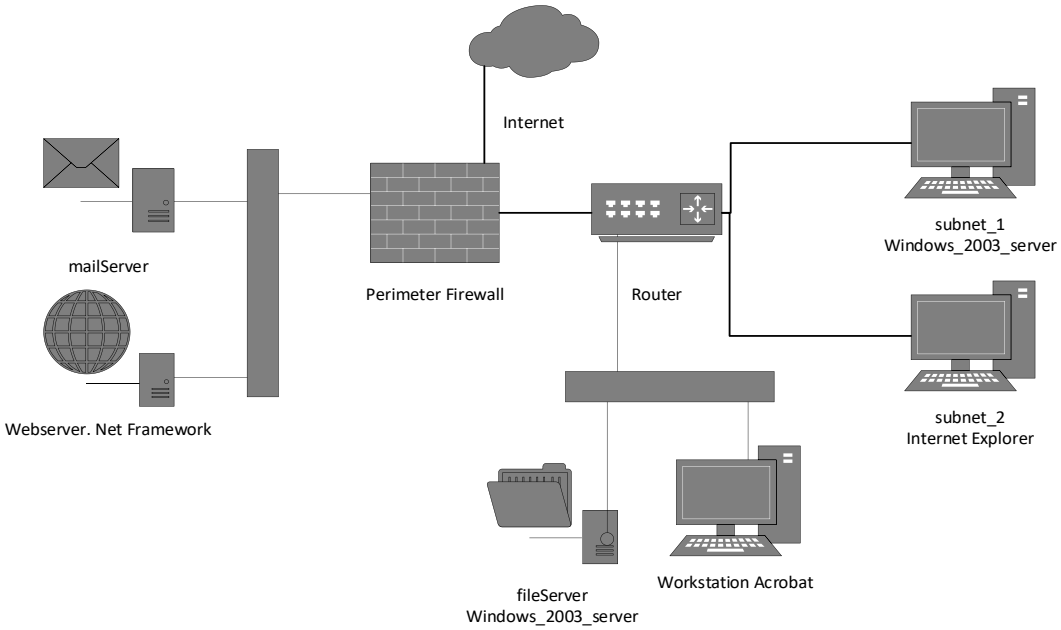

Fig. 6. Topological structure of the investigated computer network

4. The output of the file server of the first and second subnets to the external network is carried out using the HTTP protocol.

5. The file server and workstation are connected via the NFS protocol.

The studies show that this topology is characterized by several software vulnerabilities. For example, according to experts, one of the most dangerous web server vulnerabilities (CVE-2020-1198) is a buffer overflow vulnerability in the mod_uwsgi module. An attacker can cause a denial of service and execute arbitrary code.

Clients on the first subnet could be vulnerable to attack by an attacker due to the CVE -2016-0189 vulnerability in the known Vbscript.dll component. This vulnerability allows a hacker to expose arbitrary code. Clients of the second subnet are susceptible to remote code discovery attacks in the scripting engine supplied with the Internet Explorer browser (vulnerability CVE-2020-1380). The file server contains the CVE-2010-0492 vulnerability, which is associated with Windows 2003 SP 2. In this case, an attacker has the ability of implementation an attack bypassing the limitations of the original IPV4 address. Table 1 shows the indicated vulnerabilities of the system under study.

Table 1 – **Vulnerabilities of the studied system**

Hardware	List of vulnerabilities
Web server	CVE-2020-1198
First subnet	CVE -2016-0189
Second subnet	CVE-2020-1380
File-server	CVE- 2010-0492

Using the Mulval system, we will generate a graph of attacks on the studied computer system. Fig. 7 shows the generated attack graph. As shown Fig. 7 using Mulval, a graph scheme with three different types of vertices is formed. The vertices shown as rectangles characterize the configuration of the system. The diamond vertices represent potential privileges or access that an attacker could gain on the system. Elliptical nodes associate preconditions with postconditions.

The next step in the research was the procedure for simplifying the attack graph. Fig. 8 shows a refined attack graph using algorithms developed by the authors of the research [11].

The graph shown in Fig. 8 can be based on modeling using the method of deep reinforcement learning. In this figure, the location of the agent, vulnerabilities and targets of the attacker agent are represented as vertices, and each edge of the graph illustrates the capabilities of the agent and its actions. It is assumed that an attacker can move between nodes in any direction of the graph until he reaches one of the target states. It should be noted that in the developed automated penetration testing method, the general vulnerability rating system (CVSS) is also used to determine rewards in accordance with expression 1.

If a malicious agent exploits a vulnerability, then the overall score associated with that vulnerability will be considered a reward value. If the agent ultimately achieves the goal, then the reward will be considered the maximum. Additional looped edges are added to the vertices of the target with a reward of 100, since the agent who reaches the malicious target remains there forever. The rewarded DQL model is shown in Fig. 9. As an example, let us simulate a situation when a malicious agent is located at one of the points of a computer network. In Fig. 5. this point is denoted by node 12. This agent has the ability to move to other nodes (13, 21, 25 and 43), while each of his actions receives a bonus in accordance with the metrics of vulnerabilities.

Let the agent perform malicious actions starting from node 43. In this case, there are two options:

- go to node 34, which is the ultimate target of the cyberattack. In this case, the agent receives 100 points;
- return to the starting position and do not receive additional points.

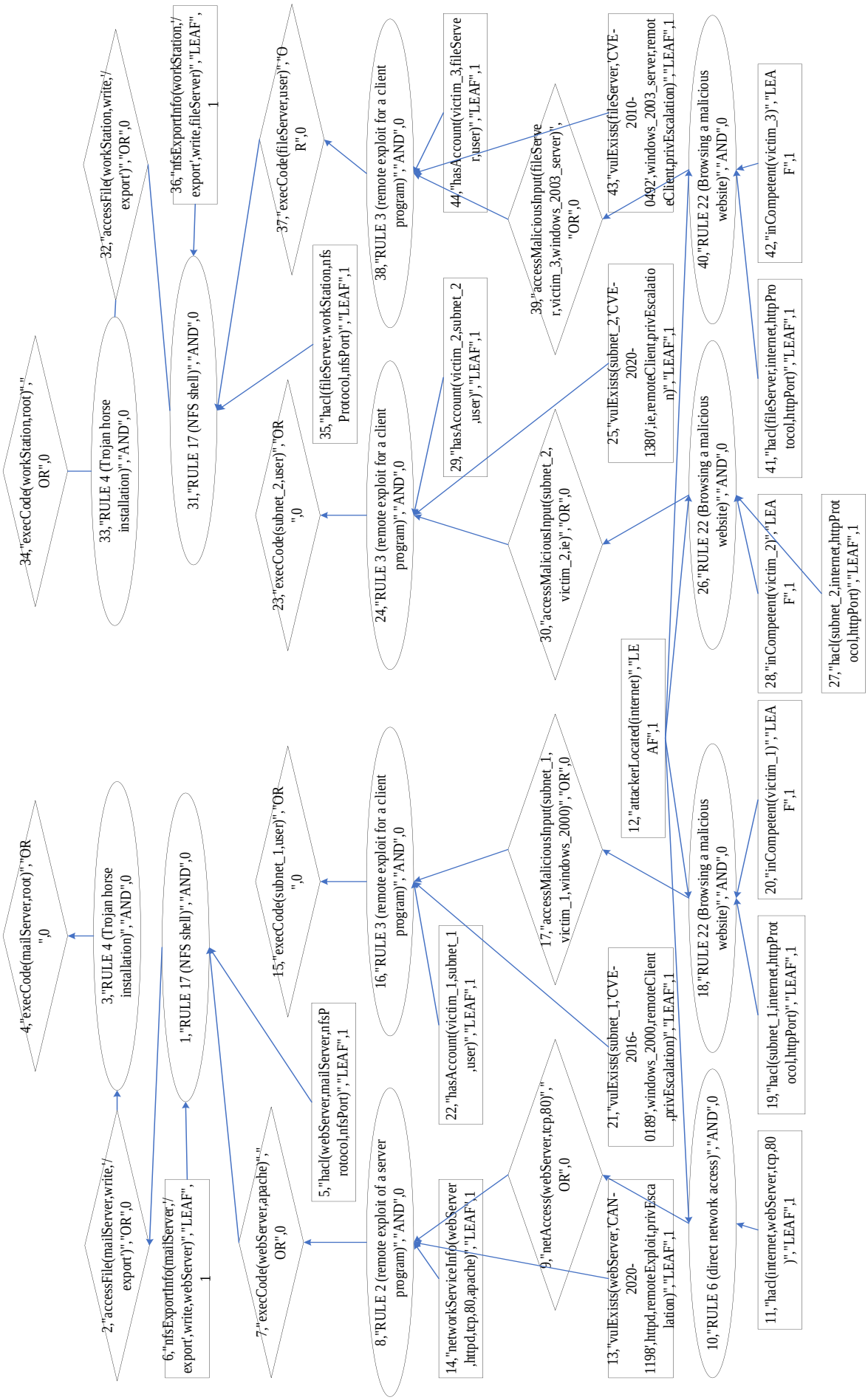

Fig. 7. Attack graph

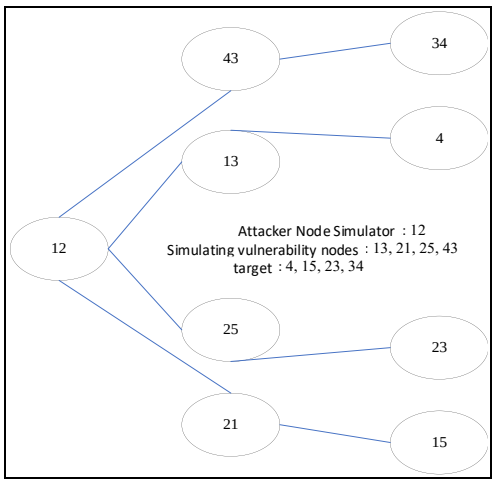

Fig. 8. Refined attack graph

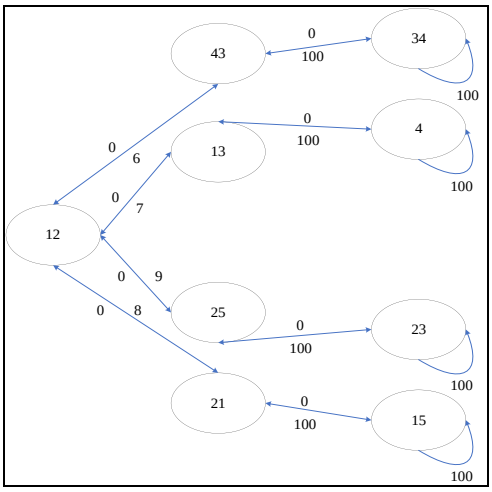

Fig. 9. Rewarded DQL Model

Let's form a generalized matrix of states in the form of a table 2. Table 2 rows represents the state of the system columns - actions. For example, matrix point 12, 43 illustrates the transition of an agent from node 12 to node 43, while receiving bonuses in the amount of 6 points. The algorithm for assessing the attacker's behavior is presented in the form of a flowchart in Fig. 10. It should be noted that the state matrix transition graph is used in this algorithm. For the purpose of practical implementation in accordance with the algorithm in Fig. 10 from the state matrix table 2 a submatrix of state clarifications and additional bonuses to the attacking agent is formed, which can be illustrated in the form of table 3.

Table 2 – **Generalized matrix of states**

	12	**13**	**21**	**25**	**43**	**4**	**15**	**23**	**34**
12	-1	7	8	9	6	-1	-1	-1	-1
13	0	-1	-1	-1	-1	100	-1	-1	-1
21	0	-1	-1	-1	-1	-1	100	-1	-1
25	0	-1	-1	-1	-1	-1	-1	100	-1
43	0	-1	-1	-1	-1	-1	-1	-1	100
4	-1	0	-1	-1	-1	100	-1	-1	-1
15	-1	-1	0	-1	-1	-1	100	-1	-1
23	-1	-1	-1	0	-1	-1	-1	100	-1
34	-1	-1	-1	-1	0	-1	-1	-1	100

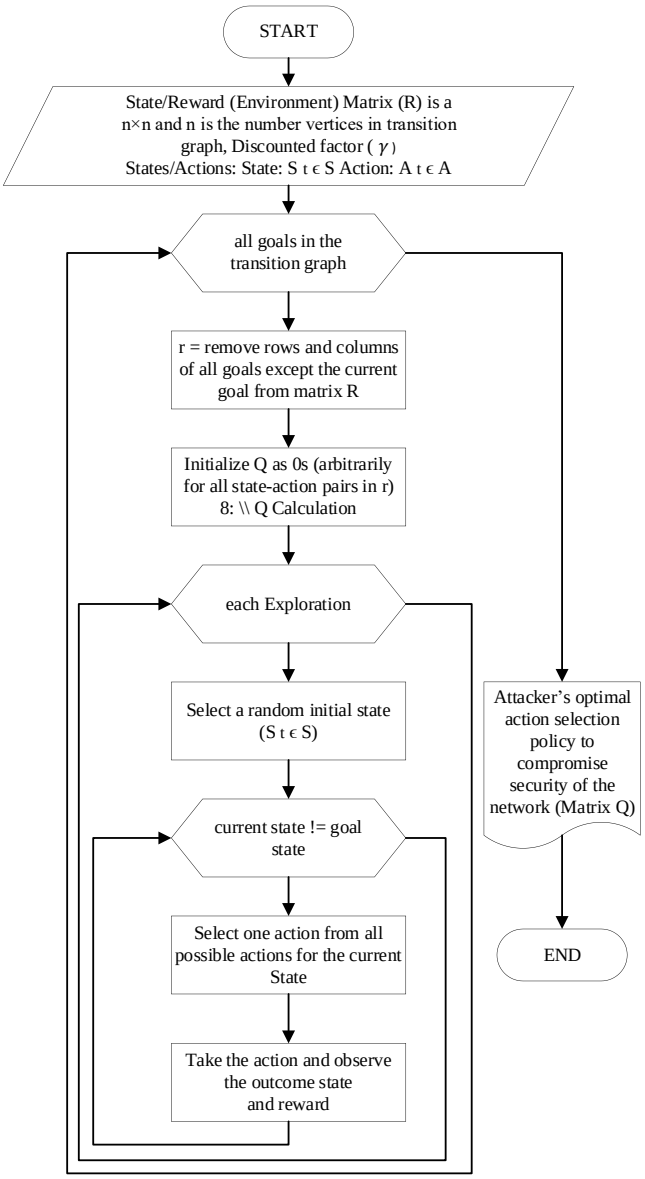

Fig. 10. Flowchart of an algorithm for assessing the behavior of an attacker

Table 3 – **Submatrix of state clarifications and additional bonuses to the attacking agent**

	12	**13**	**21**	**25**	**43**	**4**
12	-1	7	8	9	6	-1
13	0	-1	-1	-1	-1	100
21	0	-1	-1	-1	-1	-1
25	0	-1	-1	-1	-1	-1
43	0	-1	-1	-1	-1	-1
4	-1	0	-1	-1	-1	100

Using this matrix, we will form a diagram of the bonuses of the attacker agent for one of the stated goals. In fig. 11 illustrates a graph of bonuses to an agent when moving to target state 1. As shown in Fig. 11, the transition from state 12 to state 13 gives the maximum reward 408. The transition from state 13 to the state that characterizes the ultimate goal of the attacker is formalized by the maximum bonus 500. The transition 12, 13, 4 is preferable in terms of earned bonuses.

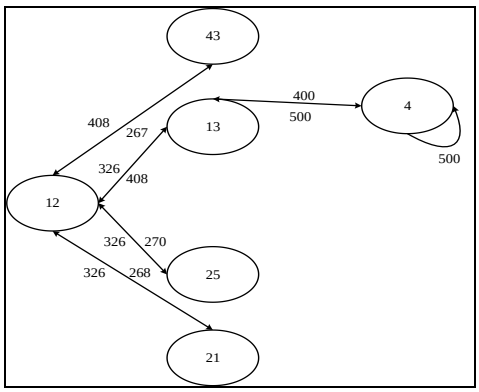

Fig. 11. The graph of bonuses to an agent upon transition to target state 1

A number of other practical examples of the results of using the developed automated penetration testing method for ethically compromising targets in a computer system are presented in Table 4. The presented table demonstrates the tester's capabilities in taking into account the maximum damage. This allows you to rank vulnerabilities in the system and eliminate them depending on their priority. Thereby it increases the efficiency of the testing process.

Table 4 – **Practical examples of the results of using the automated penetration testing method**

Attacker Node Simulator	Target	Maximum reward
12	4	908
12	15	907
12	23	906
12	34	906

To confirm the practical possibility of using the developed method of automated penetration testing, studies of the convergence of the model were carried out. This fact is clearly illustrated in Fig. 12. In fig. 12. shows a graph of the dependence of the reward for each iteration of the test performed on the number of iterations. In the experiment, 1000 iterations were considered. However, as you can see from the graph, it took 350 iterations to converge the model. The maximum reward in this example reaches 908.

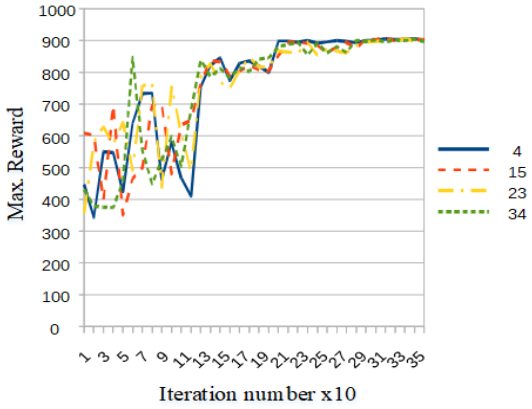

Fig. 12. Graph of the dependence of the reward for each iteration of the test performed on the number of iterations

Conclusions

A method for automatic penetration testing has been developed. A distinctive feature of the method is the integrated use of the Shodan search engine, the MulVal network security analysis platform, and software vulnerability data - CVE to obtain input data and build realistic attack scenarios and validation within the framework of deep reinforcement learning technology. This allowed generating an attack tree for various training procedures and optimizing the corresponding scripts for automated software security testing. In the study, according to the deep reinforcement learning method, the reward scores assigned to each node, according to the CVSS rating, were used. This made it possible to reduce the attack trees and identify the attack with a higher probability of occurrence. To assess the applicability of the method, an experiment was carried out and an attack tree was generated, and a testing and training scenario was also formed. The fact is confirmed that even with a small number of training scenarios, the simulation results reach 0.9 when determining the most rational attack path.

The developed method is an effective solution for software security analysis, since it allows the tester to choose a sound policy of ethical hacking and actions aimed at mitigating the negative factors of possible cyberattacks.

REFERENCES

1. Hoffmann, J. (2011), “The Metric-FF Planning System: Translating ”Ignoring Delete Lists” to Numeric State Variables”, *Journal of Artificial Intelligence Research*, vol. 20, pp. 291–341.
2. Obes, J. L., Sarraute, C. and Richarte, G. G. (1999), “Attack planning in the real world”, *Cryptography and Security*, 2013.
3. Schneier, B. (1999), “Attack trees - modeling security threats” , *Dr.Dobb’s Journal*, vol. 24.
4. Camtepe, S. and Yener, B. (1999), *A Formal Method for Attack Modeling and Detection*, URL: http://cs.rpi.edu/research/pdf.
5. McDermott, J.P. (2001), “Attack Net Penetration Testing”, *New Security Paradigms*, ACM Press, New York, pp. 15–21.
6. Ou, X., Govindavajhala, S. and Appel, A.W. (2005), “MulVAL: A logic-based network security analyzer”, 14th USENIX Security Symposium, Baltimore, MD, USA, URL: http://www.cis.ksu.Edu/~xou/publications/mulval_sec05.pdf.
7. Phillips, C. and Swiler, L. A (1998), “Graph-Based System for Network-Vulnerability Analysis”, *Proceedings of the New Security Paradigms Workshop*, Charlottesville, VA.
8. Sheyner, O. (2004), *Scenario Graphs and Attack Graphs*, Ph.D. diss., Carnegie Mellon University, Pittsburgh, PA, USA.
9. Yousefi, M., Mtetwa, N., Zhang, Y. and Tianfield, H. (2018), “A reinforcement learning approach for attack graph analysis”, *12th IEEE Int. Conf. On Big Data Science And Engineering* (TrustCom/BigDataSE), pp. 212–217.
10. Zhenguo & Beuran, Hu, Razvan & Tan, Yasuo (2020), *Automated Penetration Testing Using Deep Reinforcement Learning*, pp. 2-10, DOI: http://dx.doi.org/10.1109/EuroSPW51379.2020.00010.
11. Yousefi, Mehdi & Mtetwa, Nhamoinesu & Zhang, Yan & Tianfield, Hua (2017), “A novel approach for analysis of attack graph”, IEEE, DOI: http://dx.doi.org/10.1109/ISI.2017.8004866.
12. Sh. R. Davlatov, P. V. Kuchynski (2020) Extending the basic functionality of MALTEGO based on the canari framework and SHODAN search engine / Journal of the Belarusian state university. Physics. 2020;1:34–40

13. Madelyn, Bacon (2020), CVSS (Common Vulnerability Scoring System), URL: https://searchsecurity.techtarget.com/definition/CVSS-Common-Vulnerability-Scoring-System.
14. Riansanti, O., Ihsan, M. and Suhaimi, D. (2017), "Connectivity algorithm with depth first search (DFS) on simple graphs", *Journal of Physics*: Conf. Series, Vol. 948, ICE-STEM, Jakarta, Indonesia.

Received (Надійшла) 15.06.2021
Accepted for publication (Прийнята до друку) 18.08.2021

ВІДОМОСТІ ПРО АВТОРІВ/ ABOUT THE AUTHORS

Семенов Сергій Геннадійович – доктор технічних наук, професор, завідувач кафедри "Обчислювальна техніка та програмування", Національний технічний університет "Харківський політехнічний інститут", Харків, Україна;
Serhii Semenov – Doctor of Technical Sciences, Professor, Head of Computer Engineering and Programming Department, National Technical University "Kharkiv Polytechnic Institute", Kharkiv, Ukraine;
e-mail: s_semenov@ukr.net; ORCID ID: http://orcid.org/0000-0003-4472-9234.

Цао Вейлін – викладач інформаційного центру IT, Типовий університет Нейцзяна, Нейцзян, Китай;
Cao Weilin – teacher, Department of IT information Centre, Neijiang Normal University, Neijiang, China.
e-mail: caowl@njtc.edu.cn; ORCID ID: https://orcid.org/0000-0001-8230-5235.

Ліцзян Джан – викладач коледжу комп'ютерних наук, Типовий університет Нейцзяна, Нейцзян, Китай;
Zhang Liqiang – teacher, College of Computer Science, Neijiang Normal University, Neijiang, China.
e-mail: zhangiq@njtc.edu.cn; ORCID ID: https://orcid.org/0000- 0003-1278-2209.

Бульба Сергій Сергійович – кандидат технічних наук, доцент кафедри "Обчислювальна техніка та програмування", Національний технічний університет "Харківський політехнічний інститут", Харків, Україна;
Serhii Bulba – Candidate of Technical Sciences, Associate Professor of Computer Engineering and Programming Department, National Technical University "Kharkiv Polytechnic Institute", Kharkiv, Ukraine;
e-mail: bssserega@gmail.com; ORCID ID: https://orcid.org/0000-0003-0358-7516.

Метод автоматизованого тестування на проникнення з використанням технології глибокого машинного навчання

С. Г. Семенов, Цао Вейлін, Чжан Ліцян, С. С. Бульба

Анотація. У статті розроблено метод автоматизованого тестування на проникнення з використанням технології глибокого машинного навчання. Основна мета розробки - підвищення безпеки комп'ютерних систем. Для досягнення поставленої мети було проведено аналіз існуючих методів тестування на проникнення і виявлені їх основні недоліки. В основному вони пов'язані з суб'єктивністю оцінок в разі ручного тестування. У разі автоматизованого тестування більшість авторів підтверджують той факт, що не існує єдиного ефективного вирішення для використовуваних процедур. Це протиріччя вирішується за допомогою інтелектуальних методів аналізу. Пропонується, що розроблений метод був заснований на технології глибокого навчання з підкріпленням. Для досягнення основної мети було проведено дослідження здатності системи Shadov збирати фактичні дані для побудови дерев атак, а також платформи Mulval для генерації дерев атак. Розроблено метод формування матриці кібервторгнень за допомогою інструменту Mulval. Метод Deep Q - Lerning Network був поліпшений для аналізу матриці кібервторгнень і пошуку оптимальної траєкторії атаки. У дослідженні, відповідно до методу глибокого навчання з підкріпленням, використовувалися бали винагороди, дані кожному вузлу відповідно до рейтингу CVSS. Це дозволило зменшити дерева атак і ідентифікувати атаку з більшою ймовірністю. Проведено порівняльне дослідження автоматизованого методу тестування на проникнення. Виявлено практична можливість використання розробленого методу для підвищення безпеки комп'ютерної системи.

Ключові слова: машинне навчання; програмне забезпечення безпеки; автоматизоване тестування на проникнення.

Метод автоматизированного тестирования на проникновения с использованием технологии глубокого машинного обучения

С. Г. Семенов, Цао Вейлин, Чжан Лицян, С. С. Бульба

Аннотация. В статье разработан метод автоматизированного тестирования на проникновение с использованием технологии глубокого машинного обучения. Основная цель разработки – повышение безопасности компьютерных систем. Для достижения поставленной цели был проведен анализ существующих методов тестирования на проникновение и выявлены их основные недостатки. В основном они связаны с субъективностью оценок в случае ручного тестирования. В случае автоматизированного тестирования большинство авторов подтверждают тот факт, что не существует единого эффективного решения для используемых процедур. Это противоречие разрешается с помощью интеллектуальных методов анализа. Предлагается, что разработанный метод был основан на технологии глубокого обучения с подкреплением. Для достижения основной цели было проведено исследование способности системы Shadov собирать фактические данные для построения деревьев атак, а также платформы Mulval для генерации деревьев атак. Разработан метод формирования матрицы кибервторжений с помощью инструмента Mulval. Метод Deep Q - Lerning Network был улучшен для анализа матрицы кибервторжений и поиска оптимальной траектории атаки. В исследовании, в соответствии с методом глубокого обучения с подкреплением, использовались баллы вознаграждения, присвоенные каждому узлу в соответствии с рейтингом CVSS. Это позволило уменьшить деревья атак и идентифицировать атаку с большей вероятностью. Проведено сравнительное исследование автоматизированного метода тестирования на проникновение. Выявлена практическая возможность использования разработанного метода для повышения безопасности компьютерной системы.

Ключевые слова: машинное обучение; программное обеспечение безопасности; автоматизированное тестирование на проникновение.