One of the key processes in software development and information security management is the evaluation of vulnerability risks. Analysis and evaluation of vulnerabilities are considered a resource-intensive process that requires high qualifications and a lot of technical information. The main opportunities and drawbacks of existing systems for evaluation of vulnerability risks in software, which include the lack of consideration of the impact of trends and the degree of popularity of vulnerability on the final evaluation, were analyzed.

During the study, the following information was analyzed in the structured form: the vector of the general system of vulnerability evaluation, the threat type, the attack vector, the existence of the original code with patches, exploitation programs, and trends. The obtained result made it possible to determine the main independent characteristics, the existence of a correlation between the parameters, the order, and schemes of the relationships between the basic magnitudes that affect the final value of evaluation of vulnerability impact on a system.

A dataset with formalized characteristics, as well as expert evaluation for further construction of a mathematical model, was generated. Analysis of various approaches and methods for machine learning for construction of a target model of dynamic risk evaluation was carried out: neuro-fuzzy logic, regression analysis algorithms, neuro-network modeling.

A mathematical model of dynamic evaluation of vulnerability risk in software, based on the dynamics of spreading information about a vulnerability in open sources and a multidimensional model with an accuracy of 88.9 %, was developed. Using the obtained model makes it possible to reduce the analysis time from several hours to several minutes and to make a more effective decision regarding the establishment of the order of patch prioritization, to unify the actions of experts, to reduce the cost of managing information security risks

Keywords: risk management, information security, machine learning, vulnerability evaluation, risk scores

UDC 004.89

DOI: 10.15587/1729-4061.2021.248673

CONSTRUCTING A MODEL FOR THE DYNAMIC EVALUATION OF VULNERABILITY IN SOFTWARE BASED ON PUBLIC SOURCES

Yuliia Tatarinova
Corresponding author
Lead Engineer*
E-mail: yullia.tatarinova@gmail.com
Sinelnikova Olga
PhD, Associate Professor
Department of Algebra
and Computer Mathematics
Taras Shevchenko National University of Kyiv
Volodymyrska str., 60, Kyiv, Ukraine, 01033
Senior Engineer, Head of Laboratory
Laboratory of "DTV Security"*
*Samsung Research and
Development Institute Ukraine (SRK)
Lva Tolstoho str., 57, Kyiv, Ukraine, 01032

Received date 21.09.2021
Accepted date 20.11.2021
Published date 29.12.2021

***How to Cite**: Tatarinova, Y., Sinelnikova, O. (2021). Constructing a model for the dynamic evaluation of vulnerability in software based on public sources. Eastern-European Journal of Enterprise Technologies, 6 (2 (114)), 19–29. doi: https://doi.org/10.15587/1729-4061.2021.248673*

1. Introduction

When distributing and deploying devices and applications of the Internet of Things (IoT), companies often neglect software (SW) security issues. In the model of threats of devices of the Internet of Things, one can distinguish the applied layer of the device itself, the security of its components of system software and application packages. This space of technologies and products is quite attractive for attackers since devices can have access to personal and corporate data, infrastructure, and additional devices within the network perimeter.

The main source of information security (IS) problems is the use of an outdated code, the lack of monitoring of the security of third-party components, the lack of timely updating and processes at an enterprise that ensures the secure development life cycle (SDLC [1]). The use of components with an open-source code provides additional opportunities for finding and exploiting vulnerabilities not only for information security specialists but also for attackers. Increased risks are caused by the found vulnerabilities in the used source code packages, which are no longer supported by the authors.

Another problem is the interaction between the components in a system, namely, the existence of software dependences. The use of a vulnerable library in a program makes the latter unprotected from the actions of an attacker.

It should be borne in mind that each vulnerability carries an information security threat and a potential attack vector that can be used by attackers. A large number of vulnerabilities create high risks for enterprises and corporations. Thus, there is a growing need for rapid decision-making and taking actions to enhance information security in order to reduce the identified risks.

Negative consequences of the implementation of a threat through vulnerability carry strategic, financial, and reputational risks. It is essential to conduct vulnerability analysis to assess and mitigate risks to an organization. The complexity of systems and the growth of the level of threats lead to the need for thorough vulnerability analysis, its interaction with other components in the system, the possibility of exploitation, and the identification of consequences in the event of a successful attack. The situation is complicated by the existence of a large number of vulnerabilities in various product modules, as well as a large number of supported devices. This leads to the difficulties of timely updating, testing, and the need to carefully select the most critical vulnerabilities for further fixing. The process of vulnerability analysis, evaluation of the impact on a system, and risk anal-

ysis has a number of shortcomings and widespread problems that affect the priority of vulnerability remediation and the release of updates, namely:

– a short period of time for analysis and decision-making;

– workload of experts in information security units (in many cases there are no such units in an organization, and their functions are performed by units for testing and ensuring the functional quality of software);

– lack of comprehensive information regarding vulnerability;

– lack of up-to-date software documentation;

– insufficient resources for timely elimination of a vulnerability;

– a wide attack surface makes it difficult to identify vulnerable components.

The vulnerability spread and severity do not always correlate with actual operation capacity. Sometimes a minor bug in software can cause more harm than a critical vulnerability, especially if that error can be exploited in a chain of several similar bugs. This leads to increased security risks, as well as the expenditure of resources on more complex threat models.

In this regard, there is a need to create a system for automatic and dynamic evaluation of vulnerabilities in software. Thus, problems in the field of vulnerability analysis indicate that the research topic devoted to solving the problem of dynamic vulnerability evaluation, which helps to prioritize risk and error management, is relevant.

2. Literature review and problem statement

The most widespread and common vulnerability evaluation system is the Common Vulnerability Scoring System (CVSS) [2]. It is an open and free field standard for assessing the risks of vulnerabilities of the computing system security (OS). This method is used by MITRE [3] to assess and report detected vulnerabilities. Most companies use [3] ready-made data to assess risks for existing vulnerabilities and [2] for internal ones. The main disadvantages of this approach are the lack of context consideration, the final evaluation does not reflect the real complexity of the vulnerabilities. In practice, vulnerabilities with high risk scores are exploited much less frequently compared to vulnerabilities with low scores and a more accessible attack vector.

Paper [4] focuses on the elimination of shortcomings [2]. The study is based on the recalculation of CVSS metrics using the principal component method [5] and a change in the variance of vulnerability risk scores. However, this system is of little use for operation under real conditions, since it does not have additional knowledge about vulnerabilities, except for the vector of CVSS values, therefore it repeats the main shortcomings [2].

Study [6] provides a dynamic safety evaluation system that is characterized by improved accuracy compared to [2]. The authors note that evaluation dynamics depend on the time metrics CVSS [2], as well as on the likelihood of vulnerability exploitation. In work [7], the authors developed a vulnerability analysis system based on machine learning (decision tree) in order to prioritize the release of patches. The inputs to the model are CVSS metrics and assets characteristics based on the CVSS baseline metrics.

Studies [4–7], when solving the problem of prioritizing vulnerabilities, highlight the existence and possibility of exploitation as the main characteristic for increasing the threat level. This leads to the fact that a large number of papers are devoted to predicting the emergence of the exploitation program. Subsequent studies are devoted to solving this problem.

In study [8], the authors, using machine learning, the official dictionary of enumeration of common platforms [9], and the repository of data on vulnerability management [10], are trying to predict the emergence of a vulnerability exploitation program in the coming year. However, the model is constructed using linear regression algorithms and is based only on the existence of already published exploits, popular products and ignores the features of vulnerability trends in the IS. Thus, the model presented in [8] requires constant retraining and data updating.

Article [11] also used general information from the vulnerability database [10] and proposed a method for predicting the appearance of exploits. The main drawback of the article is the lack of sufficient data and information regarding vulnerability. The forecast accuracy is about 80 %. The authors in [11] point out that the main problem for the obtained model is an information change over time and an insufficient number of identified vulnerability features.

One of the directions of enhancing the method is the use of attack graphs. Plotting attack graphs is used both for vulnerability analysis and testing and for threat and risk evaluation. In paper [12], the NVD database and the CVSS vector were used to plot attack graphs and generate conditions and privileges for attackers using a multilayer perceptron. Research [13] provides examples of different models based on graphs of attacks, resources, and network graphs. However, the authors do not take into consideration the problems and metrics about the existence and possibility of vulnerability patches. An essential drawback is the high computational complexity. The most common use of attack graphs to calculate risk is corporate networks, rather than single computing devices.

In work [12], the authors use plotting an attack graph based on [10] to analyze the security of systems. Computational models were obtained using a database of rules and a multilayer perceptron. A feature of the study is taking into consideration the privileges of users, but a complete and accurate forecast requires more rules than was proposed by the authors.

The study that is closest to the task is [14], which presents a risk evaluation model that takes time into account. Study [14] is based on the database of exploitation programs [15], the database of vulnerabilities [16], the Panjer algorithm, and the theory of actuarial risk calculations. The main drawback of the system [14] is that it does not take into consideration the interest of the information security community, all characteristics are considered independent.

Thus, we can conclude that currently there is no unified formalized approach to analysis, evaluation, and ranking of vulnerabilities. The methods under consideration do not take into account the rate of changing input parameters and the main characteristics of vulnerabilities, which leads to the irrelevance of obtained calculations and distortion of resulting values of the IS risks. The problem of analyzing and ranking vulnerabilities affects the processes of calculating and eliminating risks, the release of a high-quality product to the market. As a result, this leads to inappropriate use of the company's resources.

According to the considered methodologies, there is a problem of the complexity of determining the significance of vulnerability impact taking into consideration the context not only of the technical environment but also the industry sector. That is why it is justified to conduct a study on the ways of enhancing the accuracy of accounting for the crit-

icality of vulnerability on the product. Such ways should also ensure risk reduction at different levels of expert's competence and the amount of source data for analysis and increase the efficiency of the software development and management process.

3. The aim and objectives of the study

The aim of the research is to develop technology and a model for automated dynamic evaluation of the impact of the vulnerability in software on the final product, taking into consideration the context of the target system. This will allow improving the quality of the final product, namely, reducing information security risks. The use of the model will minimize the time of vulnerability analysis and decision-making to remove shortcomings.

To accomplish the aim, the following tasks have been set:

– to determine the specifics, principles of operation, and amounts of information taken into consideration on the calculation of the IS risks and vulnerability management;

– to determine the main characteristics of the method of dynamic calculation of vulnerability impact on the software and the degree of their significance;

– to formalize a mathematical model for assessing the vulnerability impact based on publicly accessible sources and the relevance of the obtained information;

– to test the accuracy of the developed mathematical model on typical vulnerabilities from publicly accessible sources and databases.

4. The study materials and methods

4. 1. Description of the general architecture of the system for evaluating the vulnerability impact on the product

The general provisions of the process of extracting characteristics, architecture, and vulnerability evaluation process are the subject of research in [17]. The research [17] resulted in obtaining and formalization of a set of vulnerability features. According to the stated presentation, the original set of characteristics is represented in the form of three subsets:

$$D_i(CVE_i)=(X_i,Y_i,Z_i), \quad (1)$$

where X_i is the set of strategic vulnerability characteristics that do not depend on the target computer system and can be obtained through publicly accessible databases:

$$X_i = NVDDataProcessing(CVE_i)= \\ =(Id,CVSS_{base},Descr,V_{type},V_{treat}). \quad (2)$$

The process of searching, defining, and extracting characteristics from open sources (*NVDDataProcessing*()) is presented in research [17]. The result of the operation of *NVDDataProcessing*() function is a vector with dimensionality M_x=5 with the following parameters:

– unique vulnerability number (*Id*);

– $CVSS_{base}$ basic metrics (internal vulnerability features that are unchangeable in the user environment, consists of 2 impact scores ($CVSS_{impact}$) and exploitability ($CVSS_{exploitability}$));

– brief description of vulnerability (*Descr*);

– vulnerability type (V_{type}), which is a display of Common Weakness Enumeration (CWE);

– a type of vulnerability threat (V_{treat}).

Y_i is the feature that contains detailed information about the vulnerability and can be obtained from open sources (mainly resources such as forums, detailed reports, social networks, etc.). Some components depend on the moment of observation:

$$Y_i(CVE_i)(t)=PublicFeatureExtraction(CVE_i)= \\ =(Refs,Expl(t),Patch(t),R(t),Src(t),Trend(t)). \quad (3)$$

The algorithm of the *PublicFeatureExtraction*() function is presented in detail in paper [17]. The result of the operation of *PublicFeatureExtraction*() function is vector M_y with dimensionality 6 with the following parameters:

– the number of references and primary sources to vulnerability (*Refs*);

– availability of exploitation program (*Expl*(*t*));

– public availability of patches and updates (*Patch*(*t*));

– root cause (*R*(*t*));

– availability of open source code (*Src*(*t*));

– the value of the degree of trends and discussion of vulnerability in social networks (*Trend*(*t*)) [18].

Z_i is the feature that includes the display of features and analysis of a computer system, final values of X_i and Y_i [17]. The scheme of extraction of the basic feature is shown in Fig. 1.

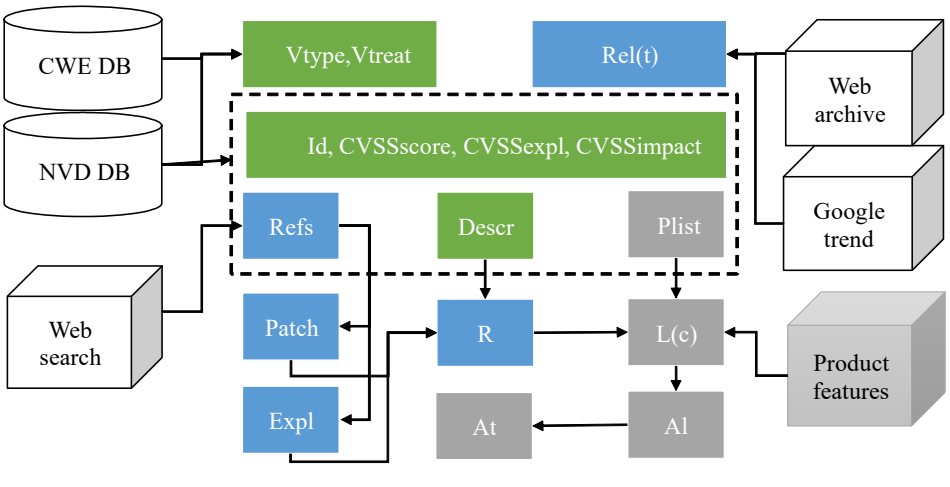

Fig. 1. General architecture of the system of vulnerability features extraction relative to a particular computing system

The units that are not taken into consideration in this study are marked in gray; green indicates the characteristics of X_i set; parameters that refer to Y_i and their derivatives are indicated in blue. The purpose of the study is to analyze the parameters and construct a mathematical model that makes it possible to get a dynamic evaluation on sets of features X_i and Y_i.

The value of the vulnerability evaluation ($E(CVE_i)(t)$) is determined by the output value of the dynamic multifactor model, to the input of which the parameters of the sets of features X_i and Y_i are sent:

$$E\left(CVE_i(t),\overline{A}\right)=Evaluation\left(X_i,Y_i(t),\overline{A}\right)=$$
$$=\begin{pmatrix} CVSS_{base_i},CSS_{\text{exploit}_i},CVSS_{impact_i},Trend_i(t), \\ Exploit_i(t),Patch_i(t),R_i(t),V_{type_i},V_{treat_i} \end{pmatrix}, \quad (4)$$

where $\overline{A}$ is the vector of parameters of the statistical model which need to be determined.

The *Evaluation*() function determines a multifactorial model and is a multidimensional convolution of vulnerability features. For a number of vulnerabilities, there are values of vulnerabilities impact $\tilde{E}(CVE_i)$. evaluated by experts The process of plotting the *Evaluation*() function boils down to building such a multifactorial model, the results of which would minimally deviate from those set by experts, according to the metrics under consideration. The parameters $\left(\overline{A}\right)$ and structure of the model (S) are determined based on the following optimization problem:

$$M\left(E\left(CVE_i(t),\overline{A}\right)-\tilde{E}\left(CVE_i\right)\right)\rightarrow\min_{\overline{A},S}, \quad (5)$$

where M are the following metrics: Mean Absolute Error (MAPE), Root Mean Square Error, the root of Root Mean Square Error (RMSE), determination factor (R^2); $\tilde{E}(CVE_i)$ the value that was obtained when analyzing the vulnerability impact by experts.

The search for parametric and structural optimization of the model is carried out based on various approaches described in chapter 4. 4.

4. 2. Source data processing and analysis of parameters to form a mathematical model for vulnerability impact evaluation

A significant part of previous studies aimed at determining the main features and characteristics of vulnerability to form the evaluation method [17]. To substantiate the accepted factors affecting the overall evaluation, it is necessary to analyze the input parameters.

In the course of early research [17], a dataset on vulnerabilities was formed based on the database [10]. The results obtained in [17] are input data for the current study. The dataset (D) contains 42835 vulnerabilities, each of which is represented as a characteristic vector. Each vector is a set of M=20 features. The vulnerabilities that were used to form the dataset were published between 2016 and 2019.

The next step was to represent all the values of the vector in numerical form. Since features V_{type}, V_{treat}, $R(t)$ are categorical and nominal, represented by a line value, it is necessary to apply one-hot encoding for them. Each value is converted into a vector that contains 1 and 0, depending on the existence of a possible category. *Id* is ordinal data, is not used as a feature in training, so it is converted to an interval type and normalization is performed.

Thus, the matrix of the following form was obtained:

$$D=\begin{pmatrix} X_1 & Y_1 \\ \dots & \dots \\ X_N & Y_N \end{pmatrix}=$$
$$=\begin{pmatrix} Base_1,Expl_1,\text{Im}\,pact_1,Trend_1, \\ Exploit_1,Patch_1,R_1,CWE_1,Treat_1 \\ \dots \\ Base_N,Expl_N,\text{Im}\,pact_N,Trend_N, \\ Exploit_N,Patch_N,R_N,CWE_N,Treat_N \end{pmatrix}, \quad (6)$$

where N=42835.

Also, based on vectors X_i and Y_i, the following features were additionally separated:

– the date a vulnerability was published ($Date_{published}$) and the date it was modified ($Date_{modified}$);

– each value from vector CVSS is represented as a separate feature;

– the groups of vulnerability types were introduced CWE_{group}.

The dimensionality of the dataset matrix is:

$$\|D\|=M\times N, \quad (7)$$

where $M=M_x+M_y+M_{extended}$, and N=42835.

Before drawing conclusions about each vulnerability, the initial dataset (D) must be analyzed and relationships, correlations, and other regularities between the presented characteristics must be determined.

To solve the problem of studying the relationship between the features and determining the linear relationship in matrix D, the matrix of Pearson correlation coefficients was calculated. Further, based on analysis of the obtained values of the correlation matrix, the most significant parameters were selected.

The next step in determining the main relationships between the characteristics was to build associative rules using the Apriori algorithm [19]. This algorithm makes it possible to find the most common sets of values of features and extract rules from them, to obtain a vector of the main factors of impact on the objective function. Based on this vector, the essential CVE_i variables are selected.

4. 3. Generation of a dataset to train the model

The next step is to form the training and test datasets to further search for a mathematical model of vulnerability impact evaluation $E\left(CVE_i(t),\right)\overline{A}$. The dataset from formula 6 of chapter 4. 2 was taken as the source set.

The main problem for further research is that dataset (D) is not marked. To solve the problem of automatic evaluation, we need a qualitatively and evenly marked set of input data. Marking the full volume with the help of expert evaluation is a very time-consuming and resource-intensive task. During data analysis, it was noticed that in the initial set, there is a duplication of data among the vectors at the exception of temporal characteristics (*Trend*, $Date_{published}$, $Date_{modified}$). The number of unique vectors after removing duplicates without taking into consideration the values of time characteristics is 9528, which greatly simplifies the task

of marking. Next, the characteristic vectors are standardized by removing the mean value and scaling to the variance of random magnitude.

A subset of vulnerabilities was selected from the initial dataset and evaluated by invited experts in the area of information security. Based on the expert evaluation, the expected values of vulnerability impact for the selected subset were determined.

Given that the number of features and the size of the dataset are still quite large, and the data were marked using expert evaluation and required a significant amount of time, it is necessary to perform cluster vulnerability analysis. This analysis will make it possible to assess the distribution of vulnerabilities, scale the dataset marked by experts and transfer the evaluation for the values that fell into the nearest cluster. To do this, it was proposed to use t-distributed Stochastic Neighbors Embedding (t-SNE). Since the size of the original dataset is large enough, to fully visualize it, it was decided to form a uniformly distributed sample of vectors (T_i) of 1000 in size. This number of vectors is sufficient to display all possible groups and clusters of vulnerabilities. The next step is to mark 10 % of vulnerabilities for each set (T_i).

4. 4. Analysis of methods of multifactor modeling

When searching for a suitable method for estimating and forming a model taking into consideration the above data and parameters, the following approaches and technologies for multifactor modeling were studied. The search for an effective evaluation method was carried out based on experimental studies and subsequent analysis of results.

4. 4. 1. Neuro-fuzzy systems

In paper [20], a fuzzy model of the *Evaluation*() function (4) was constructed, membership functions were determined and a rule base was formed based on expert evaluation. According to paper [20], the size of the characteristic vector and of linguistic variables is $len(T_i)$=8, and the number of term-sets is $len(T_ij)$=34. In order to take into consideration most possible cases, the size of the required rule base should be ≈82.000. The generation of so many rules manually is a time-consuming and inefficient problem. In this case, the characteristic vector (CVEi) consists of M=20 features.

Given the specifics of the processed data, the actual solution is to automate the generation of fuzzy rules using a fuzzy neural network, namely an adaptive network of fuzzy inference system (ANFIS). It is a five-layer neural network with a direct signal spread that implements the fuzzy Sugeno-Takagi system.

In this case, it was proposed to construct a simple Sugeno-Takagi controller with 20 inputs and 1 output. Thus, the *Evaluation*() function was represented by the procedure of fuzzy rules inference.

4. 4. 2. Construction of a multi-factor model of vulnerability evaluation

Several algorithms used to construct a model for vulnerability impact evaluation taking into consideration dynamic variables are represented below.

Linear regression. The use of such basic algorithm as multiple linear regression will not only find a connection between the input variables of the vulnerability vector ($CVE_i(t)$) from the dataset (D) and the output evaluation of vulnerability impact (*Evaluation*()) obtained by experts $\tilde{E}(CVE_i)$, but also plot the best match line for evaluation prediction $E(CVE_i)(t)$. To apply the multiple regression method to construct function $Evaluation\left(X_i, Y_i(t), \bar{A}\right)$, use the following procedures. Initially, the search for a multifactor model is made on a full set of characteristics, such as:

$$\begin{aligned} E\left(CVE_i(t), \bar{A}\right) &= Evaluation\left(X_i, Y_i(t), \bar{A}\right) = \\ &= A_0 + A_1 \times Date_{published_i} + A_2 \times Date_{modified_i} \\ &+ A_3 \times Base_i + A_4 \times Exploitability_i + \\ &+ A_5 \times Impact_i + A_6 \times Trend_i + A_7 \times Exploit_i + \\ &+ A_8 \times Patch_i + A_9 \times Rootcause_i + A_{10} \times CWE_i + \\ &+ A_{11} \times CWE_{group_i} + A_{12} \times Treat_i + A_{13} \times AV_i + \\ &+ A_{14} \times AC_i + A_{15} \times PR_i + A_{16} \times UI_i + A_{17} \times S_i + \\ &+ A_{18} \times C_i + A_{19} \times I_i + A_{20} \times J_i + \varepsilon, \end{aligned} \quad (8)$$

where $\bar{A}$ is the parameter of the model, ε is the error of the model $(\varepsilon \sim N[0,1])$.

On dataset (D), the model $\left(E\left(CVE_i(t), \bar{A}\right)\right)$ is parametrically identified by the least square method. For the resulting model $E(CVE_i)(t)$, we calculated the determination factor, which allows us to draw a conclusion about the accuracy of the model. Based on the analysis of errors, it is possible to conclude about the adequacy of the model (mathematical expectation of errors is calculated and distribution is analyzed). In this case, the errors are calculated as follows:

$$\varepsilon_i = E\left(CVE_i(t), \bar{A}\right) - \tilde{E}\left(CVE_i\right). \quad (9)$$

Initially, the model is based on the entire set of parameters, then, based on analysis of the values of $\bar{A}$, parameters, only significant values are left based on the specified threshold. The next step is to recalculate the model with a smaller set of features on the original dataset (D):

$$CVE_i(t) \to \bar{CVE}_i(t), \quad (10)$$

where the dimensionality of original characteristic vector CVE_i was M, and dimensionality of $\bar{CVE}_i$ – M', in this case $M \geq M'$.

Thus, the desired type of a multifactor model is derived experimentally at the best value of error analysis.

Polynomial regression. In order to overcome the lack of accuracy of the model, it is necessary to consider the option of applying polynomial regression. To do this, the model $E\left(CVE_i(t), \bar{A}\right)$ is sought on the class of polynomial models of the 2nd and more degrees.

Multilayer perceptron. To construct a multidimensional function of impact evaluation *Evaluation*() on the declared dataset (D) and the input characteristic vector ($CVE_i(t)$), we used a direct spread neural network. The high connectivity of the network and the nonlinear function of the activation of each neuron of a multilayer perceptron ensure computing power, and the hidden layers allow the network to extract the most important features from the input characteristic vector. When constructing a network model, one must try out the following changes to the network architecture parameters:

– activation function for the hidden level (logistic, sigmoid activation function in the form of hyperbolic tangent, semilinear element (ReLU));

– optimizer (quasi-Newton methods, stochastic gradient descent, method of adaptive assessment of moments (Adam));

– network structure (the number of hidden layers and the number of perceptrons of each layer).

Decision tree. When searching for the most accurate multidimensional *Evaluation*() model, the algorithm for constructing decision trees was considered. One of the advantages of using decision trees is data systematization and structuring when solving a problem. Thus, decision-making is performed analytically, and the output variable is built based on inductive rules.

The tree structure is optimized according to the Minimal Cost-Complexity Pruning criterion. To do this, one needs to determine the optimal value of the regularization constant (α). To do this, it is necessary to construct a tree sequence with an increase in the number of α and to select a tree with minimal error on the test sample.

Random forest. Using the decision tree algorithm with a large number of nodes and depth on the training sample provides a flexible model that tends to retrain by fitting nodes to the training dataset. As an alternative to using a tree with limited depth or determining the optimal value of regularization constant, it is possible to use a mechanism from a set of decision trees – a random forest.

5. Results of construction of a model of dynamic vulnerability evaluation

5. 1. Analysis of automated systems for calculating information security risks and vulnerability management

Table 1 gives the results of a comparison of the main existing solutions regarding vulnerability analysis and management and calculation of information security risks.

Existing systems for vulnerability management and calculation of information security risks in the system in most cases are commercial solutions. The key features of the solutions are proprietary risk evaluation systems, integration with CI/CD methodology software. The main shortcomings include the configuration and management complexity, non-transparency of the operation mechanism and used resources of vulnerability database, lack of consideration of the degree of trends, and mentions of vulnerabilities in social networks.

5. 2. The result of analyzing and determining parameters for the formation of a mathematical model of vulnerability impact evaluation

Fig. 2 shows the results of calculations of correlation analysis of vulnerability characteristics based on NVD [10] for dataset (D) and $CVE_i(t)$ characteristics extracted from the analysis of open sources.

According to the correlation matrix in Fig. 3, the following features can be identified. Sufficiently high correlation factors between the following characteristics:

– vulnerability publication date ($Date_{published}$) and the modification date ($Date_{modified}$) – 0.75;

– basic components of CVSS evaluation, namely, ($CVSS_{base}$, $CVSS_{impact}$, $CVSS_{exploitability}$);

– constituent variables of the CVSS vector (confidentiality (C), integrity (I), accessibility (A)) and basic vulnerability evaluation (*Base*);

– constituent variables of the CVSS vector (confidentiality (C), integrity (I), accessibility (A)), and metrics of impact of successfully exploited vulnerability (*Impact*).

This matrix shows the degree of the interrelation of the selected characteristics. The total sample size is 42836 vulnerabilities. The significance of correlation factors is determined by the degree of confidence of the values presented in the cells of Table 2.

By applying the Apriori algorithm, we obtained support and confidence values for the rules. The total number of rules at minimum values of support (*Support*) 0.15 and confidence (*Confidence*) 0.8 is 1325. The results of the construction and selection of the main associative rules with maximum support and confidence values are shown in Table 2.

The first column shows the most significant features, the second one shows their values and constructed rules.

Features $\{Date_{published}, Date_{modified}, CVSS_{impact}, CVSS_{exploitability}, CWE_{group}\}$ were excluded from the rule generation process due to the uselessness of time characteristics for this algorithm, as well as due to the high correlation between the constituents of the CVSS vector.

The resulting rules can be conditionally divided into 2 categories:

– high support values ($Support>0.2$) and low values of lift force ($Lift<1.2$);

– low support values ($Support\in[0.03, 0.08]$) and high values of lift force ($Lift\in[1.2, 9.91]$).

Table 1

Comparison of the most popular solutions on vulnerability management

System/Criteria	Rapid7 InsightVM/Nexpose [21]	Tripwire IP360 [22]	Tenable.io Lumin [23]	Qualys Vulnerability Management [24]
Reporting form	Web format, proprietary metrics and risk scale from 1 to 1000	Detailed vulnerability prioritization (CVSS, proprietary algorithm)	Web format, proprietary metrics of risk evaluation	Web format, pdf, csv
Integration into process	REST API, Atlassian Jira, ServiceNow, CI/CD	API, technical support	Available	API, CI/CD
Configuration and control	No internal support	High configuration complexity	Simple interface, high support requirements	High complexity of web-board control
Functional/specific features	Cloud, container and network infrastructure	Local, cloud, container resources; continuous analysis of the attack surface	Analysis with consideration of business context	Internal and cloud infrastructure, asset management capabilities, continuous monitoring
Source of vulnerability data	NVD BD	Data unavailable	Wide vulnerability base (>65,000)	Data unavailable
Product analysis degree	Scanning and analysis using Metasploit, Sonar, Nexpose	Vulnerability map with exploitation and threat tracks	Scanning of devices based on Nessus Professional	No data available, high accuracy of scanning of network devices is declared

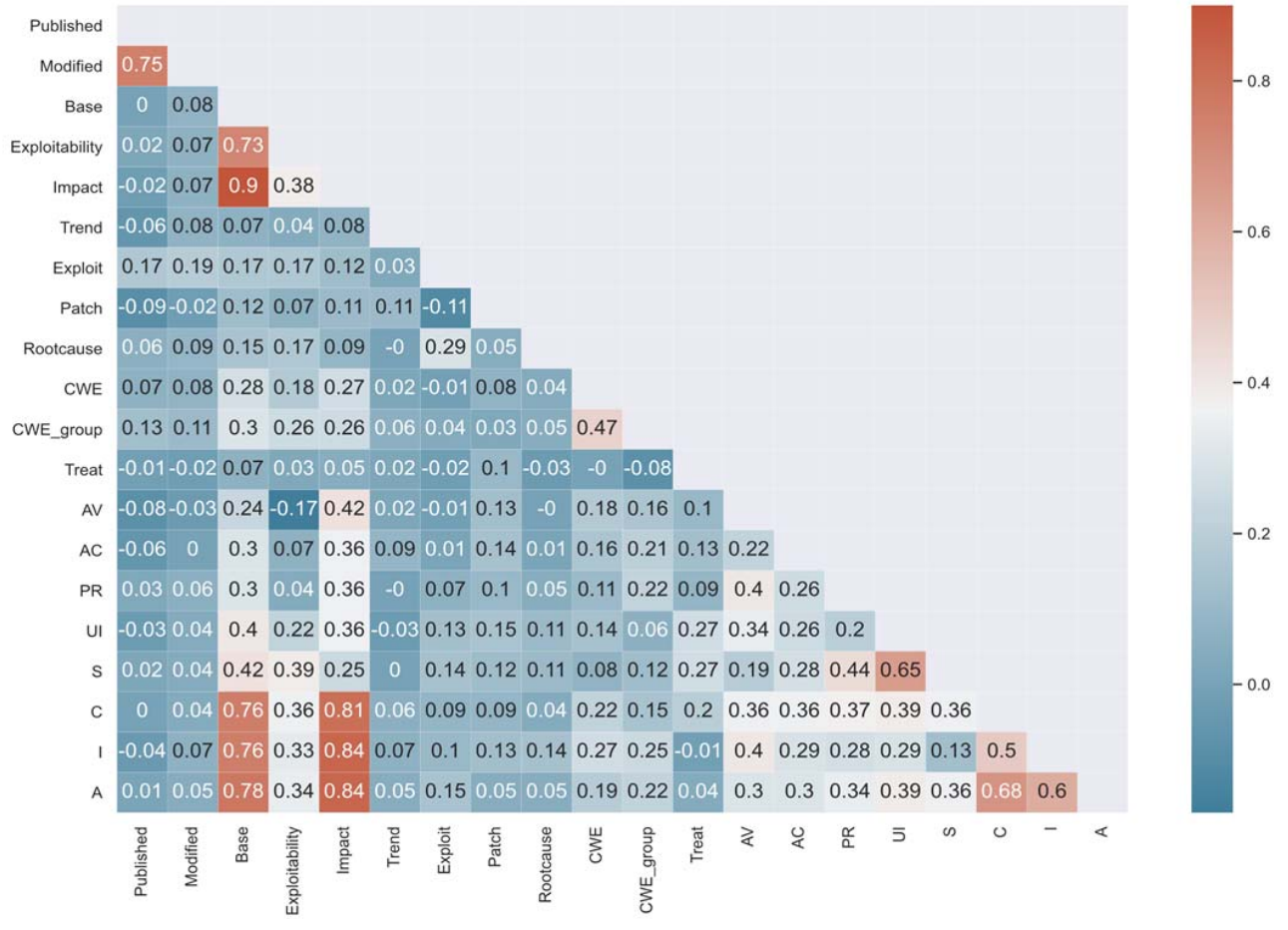

Fig. 2. Matrix of correlation of the main vulnerability characteristics

Table 2

Basic rules of relationships between features

Feature	Inferences and rules
AC	[AC=low]
AV	[AV=network, CVSS∈{9.8, 7.5, 7.6}]; [AV=local, CVSS=8.9]; [AV=physical, CVSS ∈ [3.9; 7.5)]
PR	[PR=none]
Treat	[Treat=CodeExec, CVSS∈{6.1, 5.4, 8.8, 9.8}]; [Treat=Dos, CVSS∈{7.8, 7.5, 9.8}]
Exploit	The distribution by CVSS coincides in frequency with the main number of vulnerabilities CVSS={9.8, 7.5, 8.8, 6.1, 7.8}

The first category includes the rules that are associated with the CVSS vector. At the same time, we can point out that the existence of vulnerability exploitation is associated with a low attack complexity (*AC=low*), as well as with the network attack vector (*AV=network*). Some features (for example, the existence of patches (*Patch*), trend (*Trend*)) [18] according to the obtained rules remain independent.

The second category is more interesting regarding analysis of the relationships between the features. It makes it possible to see the most pronounced relationships between the values of some characteristics, which not only reflect the state of the CVSS vector but also contain information from other sources.

5. 3. Results of construction of a model for vulnerability impact evaluation taking into consideration dynamic characteristics

Neuro-fuzzy systems. The experiment was conducted using ready-made implementations in Python (USA), as well as in the MATLAB environment (USA), namely Fuzzy Logic Toolbox (anfis) (Russia) and the interactive system Neuro-Fuzzy Designer (Russia). Using the full set of $CVE_i(t)$ variables and membership functions, both implementations were unable to handle a large amount of computation, resulting in memory shortages and subsequent errors during the network learning phase. This experiment was conducted on average hardware with an Intel Core i7 processor of the 8th generation and a RAM size of 16 GB. Reducing the characteristics of the vulnerability vector based on high correlation, as well as reducing the number of parameters of membership function, did not lead to an improvement in the results.

Table 3 gives the results of the experiment on constructing a model based on ANFIS at different sets of input characteristics.

According to the results obtained in Table 3, it can be concluded that this technology for a given dataset is not effective and cannot be applied to achieve the intended purpose.

Decision tree. When constructing a tree without limiting the parameters controlling the size of trees (maximum depth and a minimum number of samples required to divide the internal node), the tree depth is 21, and the number of leaves is 521. Metrics values: Mean Absolute Error (MAE) – 5.86; Mean Square Error (MSE) – 116.3; the Root Mean Square

Error (RMSE) is 10.78. Determination factor R^2=0.81. Based on an analysis of the resulting tree, we can conclude that the structure of the tree is complicated.

Table 3

Results of the experiment using ANFIS

Characteristic vectors	Number of term sets	Number of epochs	Mean square error	Mean absolute error
Full	42	Memory error		
{Base, Trend, Rootcause, Exploit, Patch}	13	10	10.06	101.36
		100	17.53	307.3
{Base, Trend, Rootcause, Exploit, Patch, AC, AV, PR}	22	Memory error and time restriction		
{Base, Trend, Rootcause}	8	10	9.05	81.96
		100	16.25	264.18
		1000	9.02	81.42

The results of comparing the accuracy of the *Evaluation*() function and the regularization constant (α) to optimize the structure of the decision tree are presented in Fig. 3.

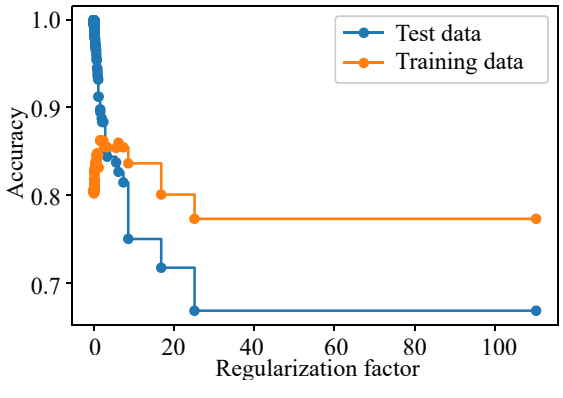

Fig. 3. Comparison of prediction accuracy (*accuracy*) and regularization factor (*alpha*(α) for training and test datasets

The greatest prediction accuracy after optimization is R^2=0.86 at α=1.6. At the same time, the number of leaves is 24, and the tree depth is 8.

Random forest. Since the size of the existing sample is not large and complete enough, training occurs with object sample with bootstrapping. The forest size is 40 trees. The results of measurements of accuracy of the random forest use exceed the accuracy of a single tree. The results of random forest testing are given in Table 4.

Linear regression. The result of the construction of a model using the full set of characteristics $CVE_i(t)$ on the dataset (D) is represented by the following equation:

$$E\left(CVE_i(t),\bar{A}\right)=0.02\times Published-0.33\times Modified+$$
$$+12.6\times Base-2.86\times Exploitability-$$
$$-4.76\times \mathrm{Im}\, pact+1.71\times Trend+5.51\times Exploit+$$
$$+2.3\times Patch-0.37\times Rootcause-1.16\times CWE+$$
$$+0.18\times CWE_{group}+0.24\times Treat+0.08\times AV+$$
$$+0.47\times AC+1.26\times PR+0.78\times UI-$$
$$-1.26\times S-0.02\times C-0.49\times I-0.56\times J. \quad (11)$$

The determination factor (R^2) is 0.895.

Taking into consideration the data obtained from correlation analysis (Fig. 2, Table 2), parametric and structural identification of the model is performed. After analyzing the obtained model and discarding insignificant vulnerability coefficients and parameters ($R(t)$, CWE_{group}, AV, C, S, J) and performing parametrical identification of the model on the original dataset, the determination factor is R^2=0.897. A decrease in the number of parameters does not lead to significant improvements in the model. The resulting linear regression equation can be written as:

$$E\left(CVE_i(t),\bar{A}\right)=-0.37\times Modified+12.87\times Base-$$
$$-3.15\times Exploitability-6\times \mathrm{Im}\, pact+1.8\times Trend+$$
$$+5.53\times Exploit+2.4\times Patch-1.04\times CWE+$$
$$+0.46\times AC+1.37\times PR+0.91\times UI-1.58\times S. \quad (12)$$

Polynomial regression. The multifactorial model for vulnerability impact evaluation uses a second-degree polynomial since the use of higher degrees (3 or more) leads to a large level of emissions in the formed sample.

Multilayer perceptron. Although the method for adaptive evaluation of moments is recommended for large datasets (1000 or more in size), this optimizer has proven to be the best in a given sample. The ReLU function turned out to be the best as an activation function on hidden layers. The model of a multilayer perceptron with three hidden layers of 30, 30, and 20 neurons in the corresponding layer was experimentally chosen as the architecture.

The obtained values of metrics of the accuracy of predicting various structures of the multifactorial model of vulnerability impact $\left(E\left(CVE_i(t),\bar{A}\right)\right)$ on the test sample are given in Table 4. The results of the values of accuracy metrics on the training sample are shown in Table 5.

Based on the data given in Tables 4, 5, it can be seen that the use of some structures of the model leads to retraining. Such methods include the application of the models constructed with the use of the algorithms of polynomial regression, random forest, as well as decision tree (with the lowest proportion of retraining).

A visualization of the results of testing the model structures and the spread of values on a test dataset are shown in Fig. 4. Blue indicates the initial values of test scores of vulnerabilities, red dots are the results of the expert evaluation.

Table 4

Comparison of the results of the construction of a multidimensional model of vulnerability impact on the test sample

Metric	Linear regression	Polynomial regression	Multilayer perceptron	Decision tree	Random forest
Mean absolute error	3.91	8.112	4.34	5.07	4.381
Root mean square error	63.94	155.182	68.92	85.54	71.279
Root from root mean square error	7.99	12.457	8.3	9.24	8.442
Determination factor	0.896	0.75	0.889	0.86	0.885

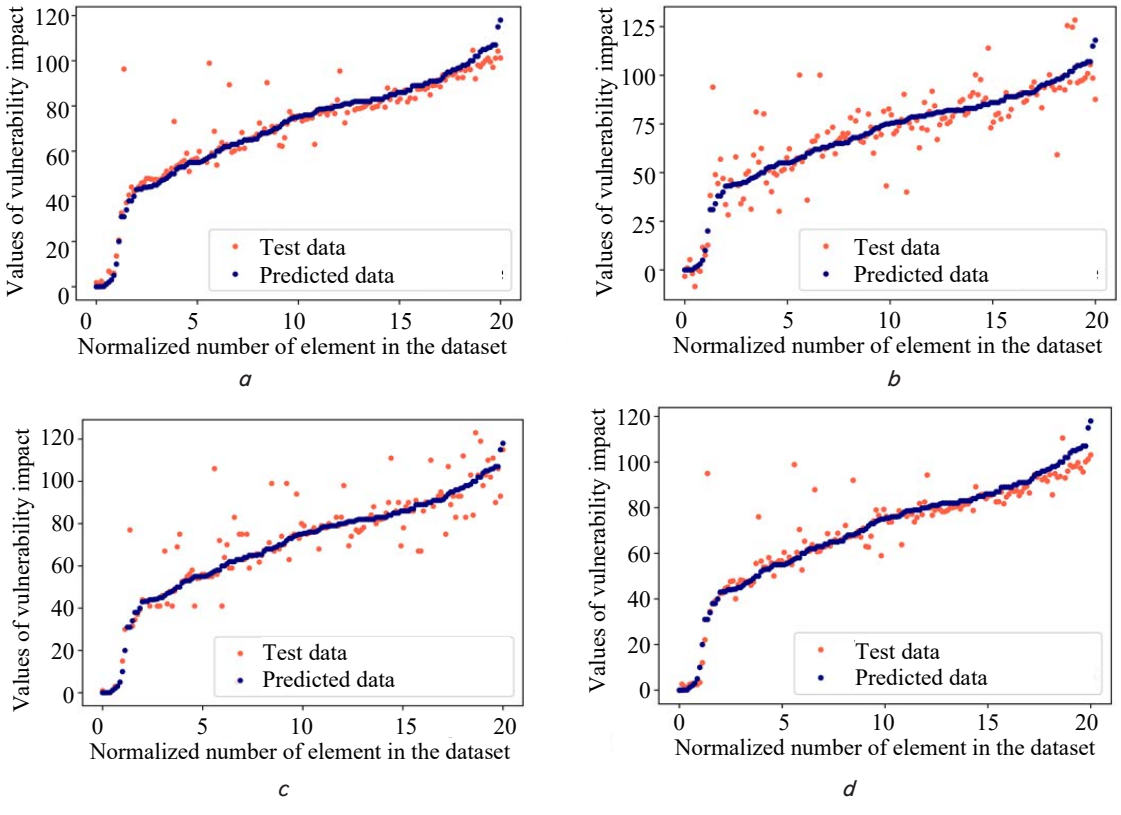

Fig. 4. Results of the model predictions on the test sample: *a* — linear regression; *b* — polynomial regression; *c* — decision tree; *d* — multilayer perceptron

Table 5

Comparison of the results of the construction of a multidimensional model of vulnerability impact on the training sample

Metric	Linear regression	Polynomial regression	Multilayer perceptron	Decision tree	Random forest
Mean absolute error	4.31	4.94	4.36	4.27	2.06
Root mean square error	96.67	66.69	90.14	52.42	18.08
Root from root mean square error	9.83	8.166	9.49	7.24	4.25
Determination factor	0.811	0.869	0.824	0.89	0.96

5. 4. Verification of the operation accuracy of the model of dynamic evaluation of vulnerability impact on actual data

The method was tested using the Python implementation (scikit-learn library) and the MATLAB toolbox (Fuzzy Logic Toolbox for experimenting with neuro-fuzzy logic).

The purpose of the experiment is to use the constructed model of vulnerability impact to visually track the change in vulnerability evaluation scores based on the changing values of assigned characteristics for the separated period.

Fig. 5 shows the result of the operation of the method of dynamic evaluation of vulnerabilities: CVE-2018-4878 and CVE-2017-1027.

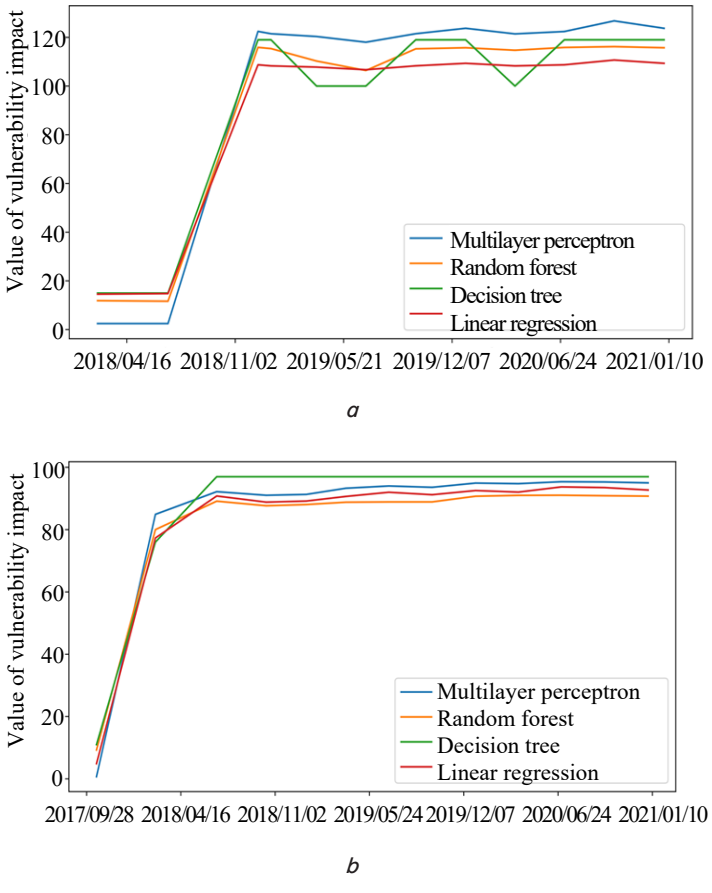

Fig. 5. Results of operation of the evaluation method: *a* — CVE-2018-4878; *b* — CVE-2017-10271

These vulnerabilities were randomly selected from the NVD database [10] for the experiment. As one can see in Fig. 6, initially, both vulnerabilities have fairly low scores due to the lack of full information after the vulnerability publication. The resulting value shows a significant increase in the arrival of additional data on vulnerability over time. Further fluctuations in the score value are affected by variable (*Trend).*

6. Discussion of the results of the development of a model for dynamic evaluation of vulnerability impact

When analyzing the parameters and studying the correlations in Fig. 3, the following conclusions can be drawn. The existence of exploitation programs (*Exploit*) has a very weak correlation with the evaluation of CVSS. At the same time, it is necessary to note the relationship between *Exploit* and vulnerability cause at the root code level (*Rootcause*). The relationship between the vulnerability trend in search engines (*Trend*) and existence of patches (*Patch*) was not detected.

Data analysis in regard to vulnerability type (CWE) shows that the main types of errors are injections (74, 78, 79), buffer overflows (119, 120, 125), and information disclosures (200) [10].

Consequently, vulnerabilities with these types will have a higher priority in developing patch recommendations.

Our analysis and the constructed set of associative rules in Table 2 make it possible to perform evaluation and generate data samples for the test and training sets, as well as for further search of a multifactor evaluation model. Stemming from analysis of the input characteristics of CVE_i (6), it can be concluded that all characteristics are significant, statistical data are poorly structured, and the dependences between the characteristics are nonlinear. Based on the obtained results in Fig. 3, it can be concluded that the existence of a trend (*Trend*) in a vulnerability is an independent magnitude.

Based on the data in Table 4, it can be seen that compared to linear regression, the root mean square deviation of the polynomial regression increased, and the R^2 indicator decreased. Thus, the use of the polynomial regression algorithm did not improve the accuracy of vulnerability impact evaluation.

As one can see in Fig. 6, the most flexible and sensitive to changes in the input data is the model based on a multilayer perceptron. This model also has some of the best prediction indicators after the linear regression model on the test sample, and this model is the least retrained. The accuracy of the obtained model is 0.889 (88.9 %).

Despite the highest prediction accuracy (89.6 %), the linear model is the least sensitive to changes in the input data.

The remaining models (decision tree and random forest), despite their high sensitivity, are over-trained and show poor results.

Therefore, it is recommended to use the model of vulnerability impact evaluation based on a multilayer perceptron. The main difference of the resulting model of vulnerability impact evaluation is that it is focused on a dynamic vulnerability risk evaluation and takes into consideration the overall vulnerability popularity, namely, trends. When training the model, the characteristics of vulnerabilities and their corresponding scores obtained by expert evaluation were used, which in turn also takes into consideration the time intervals from the publication date and the date of vulnerability modification.

The proposed model for vulnerability impact evaluation differs from the existing ones in the fact that it allows us, under conditions of limited resources and data volumes, provision of a unified and formalized vulnerability evaluation, to take into consideration current trends in the field of information security. The resulting evaluation can significantly reduce the time of basic analysis of vulnerability and make further decisions regarding its detailed study, the possibility of exploitation, or the order of applying patches.

The limitation of the resulting mathematical model is that it does not take into consideration the fading of the value of impact from the publication date. The constructed model implies that the relevance remains the same over time. The longer this interval, the less threat a vulnerability poses. In practice, vulnerabilities older than 5–10 years are rarely taken into consideration during analysis, since it is believed that the vulnerable software received an update, the source code was modified when the software functionality changed. This limitation can be eliminated by the following possible steps:

– updating the source data of expert evaluation and re-training the model (it is necessary to add to the trained sample the vulnerabilities that were published earlier, for example, 5–7 years ago);

– application of an additional coefficient reflecting the measure of removal of the current point in time from the publication date and the date of the last modification of vulnerability parameters (taking into consideration the value of trends).

Subsequently, it is planned to enhance the quality by using the following methods:

– to supplement the existing set of characteristics with the display of a cross-binary graph of a control flow for a target system and the result of its analysis;

– to expand popularity metrics through analysis in social networks and information security forums.

7. Conclusions

1. The specificity and principles of operation of existing automated systems and methodologies for calculating information security risks showed that the process of prioritizing the elimination of vulnerabilities in the development or use of software applies an insufficient amount of required resources. Most systems are limited to using a single database of NVD vulnerabilities, vulnerability evaluation metrics are closed and proprietary. The characteristics that are not taken into consideration in the evaluation were identified: separated time intervals, publication dates, the modification process, as well as the degree of vulnerability significance in the information space.

2. The main characteristics that affect the trends in changing the degree of patch prioritization were identified, namely: the existence of an exploitation program, the IS trends, certain types of vulnerabilities. Correlation analysis of all CVSS parameters, trends, types of vulnerabilities was carried out and the main relationships for dynamic calculation of vulnerability impact on software were found A significant parameter is consideration of the latest tendencies (trends) in the publication and discussion of vulnerabilities since its existence significantly increases the risk

degree and the patch importance. Vulnerabilities, the type of which is defined as injection with an access vector remote over the network, should be given the highest priority since vulnerabilities of this type are most often subject to exploitation.

3. The model of dynamic vulnerability evaluation based on open sources (NVD database, CVSS vector, vulnerability trends, existence of patches and exploits) in the form of a multilayer perceptron with a prediction accuracy of 88.9 % was developed. A distinctive feature of the obtained model is taking into consideration a change in parameters in time intervals, as well as the degree of vulnerability popularity in a specific period.

4. The model of vulnerability impact evaluation was tested on actual data, namely: on a vulnerability set from the NVD database. The developed model makes it possible to completely reduce the time during vulnerability analysis from several hours to several minutes and fully automate the process of prioritizing patches and decision-making, thus standardizing the work of experts.

References

1. Microsoft Security Development Lifecycle. Microsoft Inc. Available at: https://www.microsoft.com/en-us/securityengineering/sdl
2. Common Vulnerability Scoring System SIG. First.org, Inc. Available at: https://www.first.org/cvss/
3. Common Vulnerabilities and Exposures (CVE). Mitre.org, Inc. Available at: https://cve.mitre.org/
4. Wu, C., Wen, T., Zhang, Y. (2019). A revised CVSS-based system to improve the dispersion of vulnerability risk scores. Science China Information Sciences, 62 (3). doi: https://doi.org/10.1007/s11432-017-9445-4
5. Shlens, J. (2014). A tutorial on principal component analysis. arXiv.org. Available at: https://arxiv.org/pdf/1404.1100.pdf
6. Keramati, M. (2016). New Vulnerability Scoring System for dynamic security evaluation. 2016 8th International Symposium on Telecommunications (IST). doi: https://doi.org/10.1109/istel.2016.7881922
7. Zhang, F., Huff, P., McClanahan, K., Li, Q. (2020). A Machine Learning-based Approach for Automated Vulnerability Remediation Analysis. 2020 IEEE Conference on Communications and Network Security (CNS). doi: https://doi.org/10.1109/cns48642.2020.9162309
8. Jacobs, J., Romanosky, S., Edwards, B., Adjerid, I., Roytman, M. (2021). Exploit Prediction Scoring System (EPSS). Digital Threats: Research and Practice, 2 (3), 1–17. doi: https://doi.org/10.1145/3436242
9. Official Common Platform Enumeration (CPE) Dictionary. NIST. Available at: https://nvd.nist.gov/products/cpe
10. National Vulnerability Database. NIST. Available at: https://nvd.nist.gov/
11. Edkrantz, M., Said, A. (2015). Predicting Cyber Vulnerability Exploits with Machine Learning. Thirteenth Scandinavian Conference on Artificial Intelligence, 48–57. doi: https://doi.org/10.3233/978-1-61499-589-0-48
12. Aksu, M. U., Bicakci, K., Dilek, M. H., Ozbayoglu, A. M., Tatli, E. ıslam. (2018). Automated Generation of Attack Graphs Using NVD. Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy. doi: https://doi.org/10.1145/3176258.3176339
13. He, W., Li, H., Li, J. (2019). Unknown Vulnerability Risk Assessment Based on Directed Graph Models: A Survey. IEEE Access, 7, 168201–168225. doi: https://doi.org/10.1109/access.2019.2954092
14. Petraityte, M., Dehghantanha, A., Epiphaniou, G. (2018). A Model for Android and iOS Applications Risk Calculation: CVSS Analysis and Enhancement Using Case-Control Studies. Cyber Threat Intelligence, 219–237. doi: https://doi.org/10.1007/978-3-319-73951-9_11
15. Exploit database. Available at: https://www.exploitdb.com/
16. Vulnerability Lab. Vulnerability Research, Bug Bounties & Vulnerability Assessments. Vulnerability Lab. Available at: https://www.vulnerability-lab.com/
17. Tatarinova, Y., Sinelnikova, O. (2019). Extended Vulnerability Feature Extraction Based on Public Resources. Theoretical and Applied Cybersecurity, 1 (1). doi: https://doi.org/10.20535/tacs.2664-29132019.1.169085
18. Google Trends. Available at: https://trends.google.com/trends
19. Yuan, X. (2017). An improved Apriori algorithm for mining association rules. AIP Conference Proceedings. doi: https://doi.org/10.1063/1.4977361
20. Tatarinova, Y., Sinelnikova, O. (2019). Automatic construction of a neuro-fuzzy vulnerability risk analysis model. 2019 IEEE 14th International Conference on Computer Sciences and Information Technologies (CSIT). doi: https://doi.org/10.1109/stc-csit.2019.8929770
21. Rapid7. InsightVM. Nexpose. Available at: https://www.rapid7.com/products/insightvm/
22. Tripwire IP360. Available at: https://www.tripwire.com/products/tripwire-ip360
23. Tenable Lumin. Available at: https://www.tenable.com/products/tenable-lumin
24. Qualys Vulnerability Management. Available at: https://www.qualys.com/apps/vulnerability-management/