Processing of Unstructured Information About Software Vulnerabilities

Daniela-Kler Nkodia, Alexander Menshchikov, Dmitriy Tatarov
ITMO University
Saint Petersburg, Russia
dnkodia@itmo.ru, menshikov@itmo.ru, datatarov@itmo.ru

***Abstract*—To maintain the information security of a computer system, it is necessary to regularly audit the used system components and their corresponding vulnerabilities. However, accounting for software flaws is a time-consuming task due to the constant emergence of new unstructured information about discovered vulnerabilities. To increase the speed of analyzing relevant information, the algorithm for processing vulnerability descriptions and the XML-based output data presentation format were proposed. Processing a text description means identifying a name and version of vulnerable software, as well as determining the type of a vulnerability and a level of severity. At the current stage of the research, the achieved classification accuracy is 87%.**

I. Introduction

Every year new unstructured information about software vulnerabilities constantly appears in open sources. Considering that during the last five years the growth of the number of documented vulnerabilities is approximately 1000 per year (coming up to 20142 new vulnerabilities in 2021 [1]), the amount of undocumented information is significantly higher. Furthermore, information about detected software vulnerability initially has the form of ordinary text, which complicates the automation of data processing. Consequently, vulnerability management is an endless complex process.

To facilitate the work of security operators in understanding whether the analyzed information contains a description of the vulnerability and its type, different approaches may be applied: Artificial Intelligence, Natural Language Processing methods, and statistical analysis.

One of the solutions to determine a text related to cyber-threats is filtering by dictionaries. The idea described in [2] is to create from a text a list of terms and filter out common words that are unlikely to be related to cyber-threats. If the remaining words are present in the threat dictionary, then the text can be considered relevant. Following dictionaries may be considered as examples of non-related vulnerability words: English dictionary (contains common English words), other non-English dictionaries, stop-words dictionary (to, on, a, for, the;). Threat dictionary should contain general terms indicating known types of cyber-threats, for instance, DDoS, phishing, data breach, botnet.

Another powerful criterion to indicate information about security vulnerabilities is presence of a CVE (Common Vulnerabilities and Exposures) identifier in a text. However, this presence is a sufficient but not necessary condition. For example, the study [3] of a comparison content related to security vulnerabilities on three digital platforms: Reddit, Twitter, GitHub, analyzing conversations that include at least one CVE-ID, draw attention to the fact that some discussions begin before public disclosure. CVE registration is a time-consuming process and CVE will be publicly announced with a list of vulnerable products and versions as well as related exploits significantly later than the vulnerability detected.

To automate data processing and obtain more accurate results, Artificial Intelligence methods are applied. The following are examples of methods used:

- Among supervised machine learning methods: Naive Bayes (NB), random forest (RF), support vector machine (SVM) and logistic regression (LOG-REG), the one copes best with text classification into relevant and non-relevant is SVM [4].
- For data processing based on the words used in the expression the best machine learning method is SVM with TF-IDF [5]. The terms "remote code execution", "actively exploited in the wild", "release a proof-of-concept exploit" are always used to describe the same serious situations.
- The conference paper [6] illustrates the comparison of CNN and SVM as a hacker forum text classification method for Cyber Threat Intelligence. Accuracy of both results is high, but CNN requires more resources.
- Another group of researchers [7] also applied SVM classifier with standard unigram bag of words vector model to recognize text likely to contain security related information. For the training process they used 75 positive and 80 negative examples. Vulnerability text descriptions from NVD were used as the positive examples, and the negative examples were technical text descriptions from random websites. Preliminary evaluations (using ten-fold cross validation) on this small dataset showed that classifier was able to correctly identify all vulnerability text descriptions without using CVE-ID.
- To automatically classify exploits into pre-defined categories on the-fly the authors of [8] employed a state-of-the-art deep learning approach - Long Short-Term Memory Recurrent Neural Network (LSTM RNN). Pre-defined categories are system (RAT, keyloggers,

crypters), network (botnets, DOS), webs (XSS), database (SQLi) and mobile exploits.

- To analyze exploit source code on hacker forums researchers [9] developed a novel deep transfer learning (DTL) framework. This framework transfers the learned features from public exploit repositories to hacker forums to improve exploit labeling.

The comparison of the most efficient vulnerability description classification methods is presented in Table I. According to the quality metrics the LSTM RNN algorithm solves the classification task with the best performance.

In addition to vulnerability classification in attack types, other reviewed research [10-11] aim to determine Common Weakness Enumeration (CWE) type of vulnerability by description analysis. CWE is a list of software and hardware weakness types. Total current number of weaknesses is 918. All weaknesses are split into different groups such as cryptographic issues, data validation issues, user interface security issues, authentication errors. According to researchers from Japan University [11] assigning a CWE-ID to a vulnerability requires a precise understanding of the definition and structure of CWE-IDs. Therefore, automation would save resources and help to minimize wrong assignments by inexperienced employees.

The results of vulnerability description classification in CWE types by different algorithms are shown in Table II. The most efficient method is the combination of Boruta (the feature selection algorithm [12]) and Random Forest algorithms according to metrics from reviewed researches.

The next step of vulnerability management after the processing (classification) of discovered information is the presentation of the analysis results. Regarding studied researches, a fairly small number of works are devoted to the identification of vulnerable software and a convenient presentation of the analysis results.

In addition to the usual textual unstructured description of security-related data, there are other formalized ways to present information. Several standards were developed to share and use information about a computer system and its vulnerabilities in a more convenient way. During the research, the following standards were studied: Open Vulnerability and Assessment Language (OVAL), Software bill of materials (SBOM), Common Platform Enumeration (CPE), package URL (PURL), and CycloneDX. The use of one or a combination of the software description standards should increase the dissemination speed of the vulnerability classification result.

Thus, the purpose of this research is to increase the speed and quality of processing information about discovered vulnerabilities by applying the most accurate classification methods and providing complete analysis results in a machine-readable format. Therefore, the task of the research is to develop a new algorithm for processing unstructured information about software vulnerabilities.

TABLE I. QUALITY METRICS OF TEXT CLASSIFICATION METHODS [5], [8-9]

Classification method	Precision	Recall	F1 / Accuracy
LSTM RNN	0,970	0,980	0,980 (F1)
GRU RNN	0,960	0,960	0,960 (F1)
RNN	0,900	0,900	0,900 (F1)
SVM with TF-IDF	0,960	0,970	0,960 (Ac)
C-BiLSTM	0,876	0,872	0,874 (F1) 0,874 (Ac)

TABLE II. ACCURACY OF CLASSIFICATION METHODS BY CWE TYPE [10-11]

Classification method	Accuracy
ThreatZoom with Hierarchical Neural Network	0,930
Random Forest	0,969
Boruta and Random Forest	0,969
SVM with linear kernel	0,966
Boruta and SVM with linear kernel	0,968
Logistic Regression	0,968
Boruta and Logistic Regression	0,969
Decision Tree	0,951
Boruta and Decision Tree	0,951

II. ALGORITHM FOR PROCESSING OF AN UNSTRUCTURED DESCRIPTION

The flowchart of the proposed algorithm for processing of an unstructured information about software vulnerability is presented in Fig. 1. The input data for the algorithm is a raw unstructured text in English from a chosen source. The output of the algorithm is the structured vulnerability description in XML (eXtensible Markup Language) format, which defines vulnerable software, vulnerability type, CVE-ID, related adversarial techniques and severity level.

The algorithm consists of 10 steps:

1) Relevance check: The relevance check is performed to avoid redundant analysis. The verification is done by counting the occurrence of keywords in a text. Keywords are the chosen security related terms [13] and additionally added special words and abbreviations used to described exploits and vulnerabilities such as: injection, POC (Proof of concept), OOM (out of memory), trigger, issue. The threshold number of keywords was estimated empirically. The text can be considered relevant if there are 10 keywords for 62 words. This step completes the processing of the text defined as irrelevant.

2) Determining the name of vulnerable software: At the current stage of the algorithm development, the name is determined using a dictionary. In the future, it is planned to apply Named-entity recognition methods.

- Searching for a name using regular expressions: If none of the names from the dictionary is found, then the name of the vulnerable software is searched using regular expressions. Regular expressions look for words that contain numbers or an uppercase letter after the first letter in the word.

3) Writing down name to XML: The determined name of vulnerable software is recorded in the XML file.

4) Search for CVE-ID in text: The search is performed using the regular expression "(cve|CVE)-\d{4}-\d{4,}".

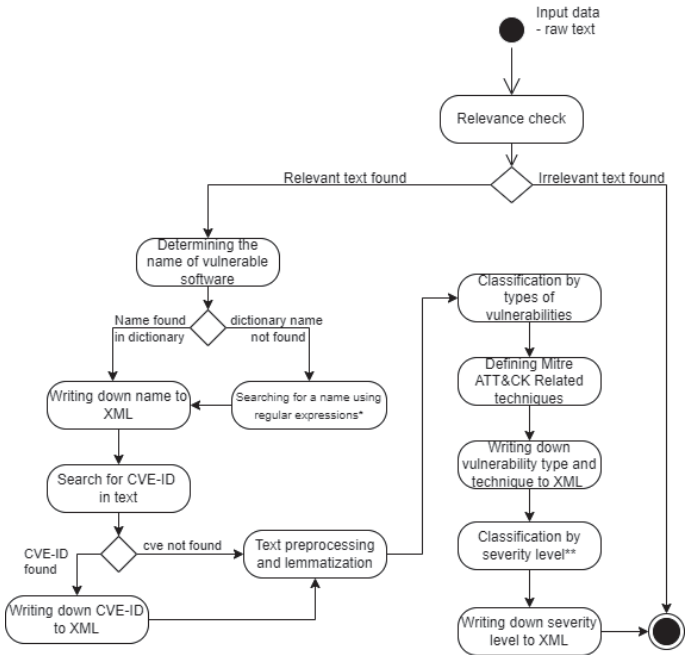

Fig. 1. Diagram of the algorithm

- Writing down CVE-ID to XML: If CVE-ID is found, then it is recorded in the XML file.

5) Text preprocessing and lemmatization: Text preprocessing consists of punctuation marks and stop-words removal, and tokenization followed by lemmatization.

6) Classification by types of vulnerabilities: After the text preprocessing, lemmas are classified. At this stage of the development of the algorithm, Support Vector Machine method with TF-IDF vectorization is used for classification by vulnerability type, since this method showed better results during experiments. For the proposed algorithm, the following 13 types were taken as classes of vulnerabilities according to the CVE details classification: Denial of Service (DoS), Code Execution, Overflow, Memory Corruption, SQL Injection, Cross-Site Scripting (XSS), Directory Traversal, HTTP Response Splitting, Bypass something, Gain Information, Gain Privileges, Cross-site request forgery (CSRF), and File Inclusion. These types are compatible with vulnerability classes by flaw type GOST R 56546-2015 standard [14].

7) Defining MITRE ATT&CK related techniques: Attack techniques are defined based on determined in previous step vulnerability type according to MITRE lookup table [15]. For example, Cross-site Scripting (XSS) vulnerability corresponds to the following techniques: T1059.007 (Command and Scripting Interpreter: JavaScript/JScript), T1557 (Man-in-the-Browser), T1189 (Drive-by Compromise), T1204.001 (User Execution: Malicious Link). Combining recently developed MITRE table with the proposed algorithm of vulnerability description classification will provide more complete representation of discovered cybersecurity flaws.

8) Writing down vulnerability type and technique to XML: The classified vulnerability type and corresponded attack techniques are recorded in the XML file.

9) Classification by severity level: Severity is an example of an additional vulnerability parameter that might be determined by the algorithm. Knowledge of severity of discovered vulnerability may help users to efficiently allocate time and resources to ensure information security. The algorithm classifies vulnerability in three groups of severity: low, medium, and high. High is the combination of high and critical groups that are used in Common Vulnerability Scoring System (CVSS). The classification is done by Support Vector Machine method. Other possible parameters for additional classification are attack vector (local, network, adjacent network), attack complexity (high, medium, low), and type of possible negative impact (violations of confidentiality, integrity, availability).

10) Writing down severity level to XML: The classified severity level is recorded in the XML file.

The proposed algorithm was implemented using Python programming language and following libraries: beautifulsoup, requests, pandas, numpy, sklearn, keras.models, keras.layers, nltk, and xml.etree.ElementTree.

III. Experiment results

Based on the analyzed studies, it was decided to use SVM method or LSTM RNN for the classification tasks in steps 6 and 9 of the proposed algorithm.

To choose the classification method and test the proposed algorithm two experiments were conducted. Data from NVD (National Vulnerability Database) and CVE details database were used for the supervised learning and algorithm evaluation process. At the current stage of the algorithm development, the quality metrics are precision, recall, accuracy, and F1-measure.

The first experiment was aimed at determining the accuracy of classification in types of vulnerabilities. A total of 13 types of vulnerabilities were used. For the neural network, it has not yet been possible to collect enough training data (classification accuracy is below 50%), nevertheless, it was possible to train the classifier based on Support Vector Machine method and obtain a classification accuracy of 87%. Table III illustrates the classification results.

The second experiment was devoted to the classification of vulnerability descriptions according to the level of severity. Based on empirical tests, it was decided to allocate three classes, combining the levels Critical and High in one. Both methods (SVM and LSTM RNN) have been trained and tested on vulnerabilities from the NVD over the past 4 years. It should be noted that the increase in the amount of data improved the classification accuracy of the method based on the neural network from 59% to 63%. Moreover, the use of more data had a positive effect on the correct classification of Log4Shell (one of the most dangerous widespread vulnerabilities in the last

couple of months) description as a high severity vulnerability. SVM method classified descriptions in levels of severity with an accuracy of 81%. The results of classification in severity levels by SVM method and LSTM RNN are presented in Table IV and Table V respectively.

TABLE III. RESULTS OF DESCRIPTION CLASSIFICATION IN TYPES OF VULNERABILITY BY SVM METHOD

Vulnerability type	Precision	Recall	F1-score
Bypass	0,900	0,900	0,900
CSRF	0,889	1,000	0,941
Directory Traversal	0,929	0.867	0,897
DoS	0,893	0,962	0,926
Code execution	0,826	0,679	0,745
File Inclusion	0,500	0,500	0,500
Http Response Splitting	1,000	0,667	0,800
Gain information	0,900	0,857	0,878
Memory corruption	1,000	0,600	0,750
Overflow	0,889	0,889	0,889
Gain Privileges	0,889	0,941	0,914
SQL Injection	0,714	1,000	0,833
XSS	0,905	0,950	0,927

TABLE IV. RESULTS OF DESCRIPTION CLASSIFICATION IN LEVELS OF SEVERITY BY SVM METHOD

Severity level	Precision	Recall	F1-score	Support
High	0,831	0,855	0,843	3493
Medium	0,778	0,767	0,773	2583
Low	0,733	0,311	0,437	106

TABLE V. RESULTS OF DESCRIPTION CLASSIFICATION IN LEVELS OF SEVERITY BY LSTM RNN

Input data	Accuracy
Based on data from 2020-2022	0, 590
Based on data from 2018-2022	0, 626

Even though Support Vector Machine method is more suitable for binary classification, it still showed better classification results than the neural network-based method. Because the latter requires more computing power and more training data.

IV. FORMAT FOR PRESENTING INFORMATION ABOUT SOFTWARE VULNERABILITIES

To develop an easy-to-use solution appropriate for sharing results of text processing, the algorithm should provide machine-readable output. Therefore, certain standards and solutions were studied.

One of the international machine-readable standards, which helps to transfer the information and to assess and report upon the machine state of computer systems, is Open Vulnerability and Assessment Language (OVAL) [16]. OVAL includes three schemas written in XML to describe a specific machine state (vulnerability, configuration, patch state, etc.), to represent system information, and to report the results of an assessment: OVAL definitions schema, OVAL System Characteristics schema, and OVAL Results schema. OVAL Definition expressing a vulnerability includes the OVAL-ID, status of the definition, the CVE Identifier or other reference on which the definition is based, version of the Schema, a brief description of the security issue covered in the definition, the main author, a list of the significant contributors to the development of the definition, the specific OS, the name of the file with the vulnerability in it, application version, and patch status. Using this information, it is easier to find out if a system is vulnerable.

Some of possible OVAL use cases are Vulnerability Assessment, Malware and Threat Indicator Sharing, Patch Management, and Auditing and Centralized Audit Validation. ScanOVAL [17] is an example of a software that automatically detects vulnerabilities of a software installed on a local PC based on processing data presented in XML files.

Other examples of vulnerability description standards are The Common Vulnerability Reporting Framework (CVRF) standard, which is also an XML-based language, and State Standards such as GOST 56545-2015 [18]. The disadvantage of these formats compared to OVAL is either that they contain less information or that they do not increase the speed of information exchange.

Another useful format for presenting information about a computer system is Software Bill of-materials (SBOM) [19]. SBOM is a list of all software packages (with version numbers) that are incorporated into the build of a software product. There are multiple SBOM standards including CycloneDX, SPDX, and SWID. CycloneDX [20] focuses on software security use cases, for example, vulnerability analysis (software and hardware). It is designed with machine-readable formats XML and JSON to be extensible, and easily adoptable. CycloneDX presents information as components with sub-components. Components are names and versions of open-source applications, libraries, frameworks, an operating system, or file. After the generation of BOMs, the components can be tracked and automatically analyzed during the software lifecycle via specifically designed applications such as Dependency-Track [21].

Dependency-Track analyzes and continuously monitors components from SBOMs for security, operational, and license risk. To identify known vulnerabilities the following sources are used: NVD, Node Package Manager Public Advisories, Sonatype OSS Index, VulnDB from Risk Based Security.

Dependency-Track is not the only application that provides information about possible vulnerabilities depends on certain SBOM. Efros Vulnerability Checker [22] based on the

category type (software, OS, virtualization system, network equipment), manufacturer name, product name and version number, displays a list of vulnerabilities sorted by criticality. Each retrieved description of a vulnerability includes ID (CVE-ID or OVAL-ID), a description, hazard level, date of detection and link(s) to known vulnerability database(s) with more detailed information: NVD, bdu.fstec, and Security Advisories from various developers.

The essential advantage of the implementation of developed machine-readable standards is the ability to share information in a single format, increasing the speed of information exchange and digestion. However, the reviewed systems cannot be called absolute solutions to provide all relevant information about possible vulnerabilities. These applications fully depend on advanced prepared and processed information, which appears after certain delay needed for the analysis.

Thereby, based on studied solutions, it was proposed to prepare a format similar to CycloneDX, including a description of the vulnerability, in order to provide convenient results of vulnerability description processing. The format is an XML-based and contains:

- Date of creation.
- Source reference.
- Name and version of vulnerable software.
- Vulnerability description.
- CVE-ID, if available.
- Vulnerability type.
- Additional parameters. For example, severity level, attack vector, attack complexity, attack technique.

The example of the filled-in XML description of LiquidFiles vulnerability is presented below.

```
<?xml version="1.0" encoding="UTF-8"?>
  <new_format xmlns="...">
     <generator>
       <schema_version>0.3</schema_version>
       <timestamp>2020-04-8_11:00</timestamp>
     </generator>
     <source>
       <title>Shares feature Weakness Allows code injection</title>
       <reference ref_url="https://threatpostxxx.com/new_vulnerability"/>
     </source>
     <components>
       <component>
         <name>LiquidFiles</name>
         <versions>
           <version>before 3.3.19 </version>
         </versions>
         <vulnerable_component>Shares feature of LiquidFiles</vulnerable_component>
         <vulnerability>
           <type>XSS(vulnerability type according to ML classifier)</type>
           <techniques>T1059.007</techniques>
           <severity>High</severity>
           <cve>CVE-2020-29071</cve>
        <description_notes>The issue arises from the insecure rendering of HTML files uploaded to the platform as attachments, when the htmlview URL is directly accessed. The impact ranges from executing commands as root on the server to retrieving sensitive information about encrypted e-mails, depending on the permissions of the target user.</description_notes>
       </vulnerability>
     </component>
     ...
  </components>
</new_format>
```

V. Conclusion

Early awareness of discovered software vulnerabilities is essential for protection. Considering that the information about newly discovered security flaws constantly appears in a noisy environment, the research was devoted to creating a solution that helps ensure well-timed defense.

The proposed method for increasing the speed of processing unstructured information is to automatically determine the name of the vulnerable software and the parameters of the new vulnerability, as well as the subsequent presentation of the data in a machine-readable form. Based on the studied research, it is proposed to apply one of the following classification methods to determine the type of vulnerability: Support Vector Machine, Long Short-Term Memory Recurrent Neural Network, and Random Forest, since their accuracy parameters surpassed others.

From the entire flow of information, the developed algorithm determines the relevant information. Then the name of the vulnerable software is automatically detected and, together with such vulnerability parameters as type, severity, attack technique and CVE-ID, is written to the developed XML-based format.

At this stage of the study, the use of two (SVM, LSTM RNN) of the three most effective classification methods has been tested. Obtained classification accuracy in vulnerability types and severity levels is 0,87 and 0,81 respectively.

The developed algorithm for processing unstructured data of software vulnerabilities can be used as an integral part of the security analysis system to reduce the company's potential costs.

The future research will be aimed at improving the classification accuracy of the developed algorithm, choosing

an appropriate method for determining the name of vulnerable software, and will be focused on conducting additional tests. In particular, it is planned to perform additional optimization of LSTM RNN parameters and conduct testing of vulnerability description classification by Random Forest machine learning algorithm. When choosing a method for determining the name of vulnerable software, Named Entity Extraction methods will be analyzed and compared with the dictionary search approach. In order to verify the applicability of the selected text relevance assessment method, tests will be carried out on a larger dataset. Also, to increase the functionality of the proposed algorithm, it is planned to test textual description classification in additional vulnerability parameters, for instance, in levels of attack complexity.

ACKNOWLEDGMENT

This work was supported by the Ministry of Science and Higher Education of Russian Federation, passport of goszadanie no. 2019-0898.

REFERENCES

[1] CVE Details, Vulnerabilities By Type, Web: https://cvedetails. com/vulnerabilities-by-types.php.

[2] A. Sapienza, A. Bessi, S. Damodaran, P. Shakarian, K. Lerman, E. Ferrara, “Early Warnings of Cyber Threats in Online Discussions” - *IEEE International Conference on Data Mining Workshops (ICDMW),* IEEE, 2018, pp. 667-674.

[3] S. Horawalavithana et al., “Mentions of Security Vulnerabilities on Reddit, Twitter and GitHub” - *IEEE/WIC/ACM International Conference on Web Intelligence*, PLoS ONE, 2019, pp. 200-207.

[4] E. Nunes et al., “Darknet and deepnet mining for proactive cybersecurity threat intelligence”, *IEEE Conference on Intelligence and Security Informatics (ISI)*, IEEE, 2016, pp. 7-12.

[5] R. E. Radu, O. Grigorescu, R. V. Rughiniș, “Security News Aggregator”, *8th RoEduNet Conference: Networking in Education and Research (RoEduNet),* IEEE, 2019, pp. 1-8.

[6] I. Deliu, C. Leichter, K. Franke, “Extracting cyber threat intelligence from hacker forums: Support vector machines vs convolutional neural networks”, *IEEE International Conference on Big Data (Big Data)*, IEEE, 2017, pp. 3648-3656.

[7] V. Mulwad, W. Li, A. Joshi, T. Finin, K. Viswanathan, “Extracting Information about Security Vulnerabilities from Web Text”, *IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, IEEE, 2011, pp. 257-260.

[8] R. Williams, S. Samtani, M. Patton, H. Chen, “Incremental Hacker Forum Exploit Collection and Classification for Proactive Cyber Threat Intelligence: An Exploratory Study”, *IEEE International Conference on Intelligence and Security Informatics (ISI)*, IEEE, 2018, pp. 94-99.

[9] B. Ampel, S. Samtani, H. Zhu, S. Ullman, H. Chen, “Labeling Hacker Exploits for Proactive Cyber Threat Intelligence: A Deep Transfer Learning Approach”, *IEEE Intelligence and Security Informatics*, IEEE, 2020, pp. 1-6.

[10] E. Aghaei, W. Shadid, E. Al-Shaer, “ThreatZoom: CVE2CWE using Hierarchical Neural Network”, *16th EAI International Conference on Security and Privacy in Communication Networks*, CoRR, 2020.

[11] M. Aota, H. Kanehara, M. Kubo, N. Murata, B. Sun, T. Takahashi, “Automation of Vulnerability Classification from its Description using Machine Learning”, *2020 IEEE Symposium on Computers and Communications (ISCC)*, IEEE, 2020, pp. 1-7.

[12] M. B. Kursa, A. Jankowski, W.R. Rudnicki, “Boruta - A system for feature selection”, *Fundamenta Informaticae*, IOS Press, 2010, vol. 101, no. 4, pp. 271–285.

[13] National Institute of Standards and Technology Interagency or Internal Report, NISTIR 7298 R2, “Glossary of Key Information Security Terms”, 2013, 222p.

[14] Federal Agency for Technical Regulation and Metrology, GOST R 56546-2015, “Information protection. Vulnerabilities in information systems. The classification of vulnerabilities in information systems”, 2015, 12p.

[15] Github repository of the project: Mapping MITRE ATT&CK® to CVEs for Impact, Web: https://github.com/center-for-threat-informed-defense/attack_to_cve.

[16] MITRE OVAL official website, Web: https://oval.mitre.org /index.html.

[17] BDU FSTEC official website, ScanOVAL program description, Web: https://bdu.fstec.ru/site/scanoval.

[18] Federal Agency for Technical Regulation and Metrology, GOST R 56545-2015, “Information protection. Vulnerabilities in information systems. Rules of vulnerabilities description”, 2016, 12p.

[19] NTIA Multistakeholder Process on Software Component Transparency, Standards and Formats Working Group, “Survey of Existing SBOM Formats and Standards”, Version 20191025, 2019, 31p.

[20] CycloneDX official website, Web: https://cyclonedx.org/.

[21] Dependency-Track official website, Web: https://dependency track.org/.

[22] EFROS Config Inspector. EFROS Vulnerability Database, Web: https://checker.gaz-is.ru/.