Topical Issues Related to Certification Tests of Information Security Tools

Vitali V. Varenitca[1,2], Alexey S. Markov [1,2] and Pavel A. Naschokin[2]

[1] *Bauman Moscow State Technical University, 5/1 2nd Baymanskay ul., Moscow, 105005, Russia*
[2] *NPO Echelon, 24 2nd Electrozavodskaya ul., Moscow, 107023, Russia*

Abstract

This paper studies various methods and techniques used to identify defects and vulnerabilities during the certification tests of information security tools. The conclusion was drawn on the relevance and priority of the examination of open source web applications. The paper cites the study and demonstrates the drawbacks of directive methods used to find the vulnerabilities and undocumented features in software products. The author's detailed statistics was provided demonstrating the identified vulnerabilities by the classes of computer attacks, information security tool manufacturers, programming environments and vulnerability identification procedures. Original test methods were compared with well-known directive test procedures. The relevance of introducing the concept of secure software development is shown. Recommendations are given on improving the security of software tools used for information protection.

Keywords

Certification tests, information security tools, vulnerability identification methods, secure software code

1. Introduction

The issue of identifying vulnerabilities in software is certainly not new, but it has not been solved definitively and is extremely topical at the present time [1]. The very presence of vulnerabilities in software creates the main class of threats in contemporary computer systems and networks (e.g. refer to [2-8]). However, the issues of vulnerability identification take on particular importance in the course of certification tests of information security tools (IST) because this procedure is mandatory [9-13]. Moreover, the vulnerability analysis of IST software (SW) is now becoming one of the main activities in the development and support of secure software products [14-21].

As for IST certification, this work is performed both during certification for compliance with the requirements of security profiles approved by the Russian Federal Service for Technical and Export Control (FSTEK), which explicitly include the requirements of trust family Vulnerability Analysis, and during the tests for compliance with the requirements of specifications or traditional regulatory documents [13]. The conceptual approach to vulnerability analysis recommended at the present time by FSTEK of Russia suggests the combined use of approaches described in ISO/IEC 18045 and ISO/IEC TR 20004 [22-24]. In general, the procedure involves the following steps [25]:

BIT-2021: XI International Scientific and Technical Conference on Secure Information Technologies, April 6-7, 2021, Moscow, Russia
EMAIL: www@cnpo.ru (A.1); a.markov@bmstu.ru (A. 2); pavel-vivos@yandex.ru (A. 3)
ORCID: 0000-0003-0111-7377 (A. 2)

1. Identification of known (confirmed) vulnerabilities of the object to be certified. During this step the experts of test laboratory search for any known (confirmed) vulnerabilities in public information sources such as the Data Bank of Information Security Threats of FSTEK of Russia or CVE list [26, 27].

2. Identification of vulnerabilities of the object to be certified not published previously. During this step the experts of test laboratory analyze the data on the object to be certified (source code, available documentation, information from open sources) to define the list of potential vulnerabilities of the object to be certified and to develop and perform the penetration test for each identified vulnerability in order to determine the assumption accuracy.

Considering that the requirements to perform the vulnerability analysis is relatively new to Russian certification systems for information security tools, currently there are almost no procedural guidelines for test laboratories which could be used to perform an effective analysis of web application vulnerabilities. This determines the relevance of the task of developing and improving the procedures for vulnerability analysis during certification tests for compliance with the information security requirements and during the assessment of compliance with the requirements of modern security standards.

2. Adapted procedure for web application vulnerability analysis

The scope of this study included the approbation of combined procedure for software vulnerability analysis based on the methods suggested in studies [25, 28] and the requirements of modern information security (IS) standards, as well as formulation of recommendations for experts certified by the test laboratory.

The figures below (Figure 1 – Figure 3) demonstrate the main stages of the suggested procedure.

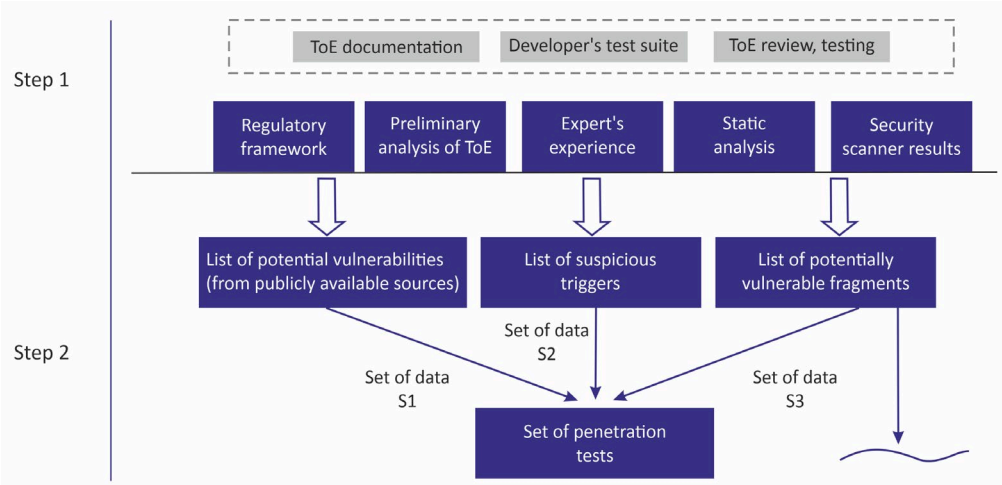

Figure 1: Stages 1 and 2 of the procedure for web application vulnerability analysis

The stages and steps of the adapted procedure for software vulnerability analysis are briefly described below.

Stage 1. Acquisition of data to perform the vulnerability analysis.

Step 1. Identification of the minimum set of initial data:

- Documentation for the assessed object;
- Test-cases developed by the manufacturer to perform internal tests at the stages of assessed object life cycle.

Step 2. Analysis of identified data. Based on the available data, the expert shall analyze the documentation for the assessed object. Examination of the documents makes it possible to understand what technologies and software tools were used to design the product under consideration and allows forming a minimum set of conditions required for correct operation of the assessed technical tool. The investigation can be divided into the components described below.

1. Search for identification attributes of assessed object.
2. Search for information about the applied third-party technical tools which are necessary for the assessed object operation.
3. Search for information about the borrowed components of the assessed object.
4. Identification of the list of the assessed object configurations and operational environments.
5. Identification of the protection mechanisms used for the assessed object.

Step 3. Examination of publicly available information sources. The expert uses the information obtained at the previous steps to analyze the open information sources (regulatory database) for availability of attack patterns for known vulnerabilities in the software components of operational environment and products similar to the assessed object, vulnerabilities in configurations that enable the assessed object operation, information about any errors during interaction of third-party technical tools with the assessed object or similar products. The analysis results serve as the basis for preparation of vulnerabilities list defined as potential vulnerabilities of the assessed object.

Step 4. Identification of the set of source texts used to compile the object to be certified. During this step, the experts of test laboratory check the software source texts presented for certification for completeness and lack of redundancy in order to define the exact set of source texts involved in the software compilation. While performing this step, the experts of test laboratory use the information generated by the building system and various tools (file system monitors, file system audit programs, etc.). The main purpose of this step is to document the list of source text files used to compile the certified object.

Step 5. Static signature analysis [30, 31] with respect to the set of source texts documented at step 4. The static analyzer used shall be able to search for potentially hazardous constructs in the source texts and form this list while assigning the CWE database identifier to each potentially hazardous construct identified.

Step 6. Preliminary analysis of the assessed object. The preliminary analysis is carried out to:

- Clarify the data obtained during the documentation examination;
- Assess the documentation submitted to perform tests for compliance with the actually studied product;
- Perform test actions that allow identifying incorrect operation of the assessed object;
- Perform the test actions directed to the assessed object which allow identifying special attributes of the assessed object demonstrating the potential presence of certain varieties of code defects (vulnerabilities);
- Define any additional undocumented ways of action on the assessed object which are potentially able to compromise the data integrity/availability/confidentiality.

The preliminary analysis allows the expert to define the product purpose more accurately and understand which of the available skills might be of use in the further product analysis, as well as to develop the higher quality penetration tests. The data acquired during the preliminary analysis can influence the results obtained during the analysis of the assessed object documentation and sets of texts provided by the developer. While examining the information available in the open sources (regulatory database) the expert shall use the data obtained during the preliminary analysis and form the additional list of potential vulnerabilities in the assessed object. Moreover, in the course of preliminary analysis the expert shall use the data obtained from the static analysis of assessed object code and the results of security scanner operation. While performing the preliminary analysis, the expert shall analyze the results of static analyzer and security scanner operation thus minimizing false operations. The expert shall use the results

of preliminary analysis and documentation analysis to develop the data required for the second stage of this procedure.

Stage 2. Preparation of the original list of potential vulnerabilities in the assessed object

Step 7. Evaluation of the data obtained at stage 1.

Step 8. Preparation of the list of potential vulnerabilities in the assessed object based on the findings of stage 1.

Step 9. Identification of suspicious operations performed by the assessed object. Assessment of identified incidents for potential impact on the web application security (e.g. [10, 25, 32, 33]).

Step 10. Preliminary fuzzing tests. This is a nonintelligent type of fuzzing tests aimed to obtain specified inputs for dynamic tests [20, 35].

Step 11. Processing of the obtained list of potentially hazardous constructs using the filtration criteria defined in section 6.1.2.1 of standard ISO/IEC TR 20004.

Step 12. Preparation of the list of attack patterns relevant for the software under study using the sequence of actions defined in section 6.1 of ISO/IEC TR 20004.

Stage 13. Creation of potential vulnerability/attack pattern pairs. Processing of the lists of potential vulnerabilities and attack patterns created at stage 2 using the sequences in section 6.1.2.2 of ISO/IEC TR 20004.

Stage 3. Dynamic analysis of the assessed object code.

Step 14. Identification of the code sections required to perform the code instrumentation.

Step 15. Writing of instrumentation and profiling functions, embedding of the instrumentation and profiling functions into the code.

Step 16. Development of the test set of input data.

Step 17. Dynamic analysis of the code. Identification of suspicious operations of the functions tested. Preparation of the list of potential vulnerabilities.

Step 18. Processing of the obtained list of potentially hazardous constructs using the filtration criteria defined in section 6.1.2.1 of ISO/IEC TR 20004. Comparison with the list created at stage 2 of this procedure.

Stage 4. Update of the penetration test set based on the completed dynamic analysis of the assessed object.

Step 19. Preparation of the list of attack patterns relevant for the software under study using the sequence of actions defined in section 6.1 of ISO/IEC TR 20004. Update of the tests obtained in step 10 of this procedure.

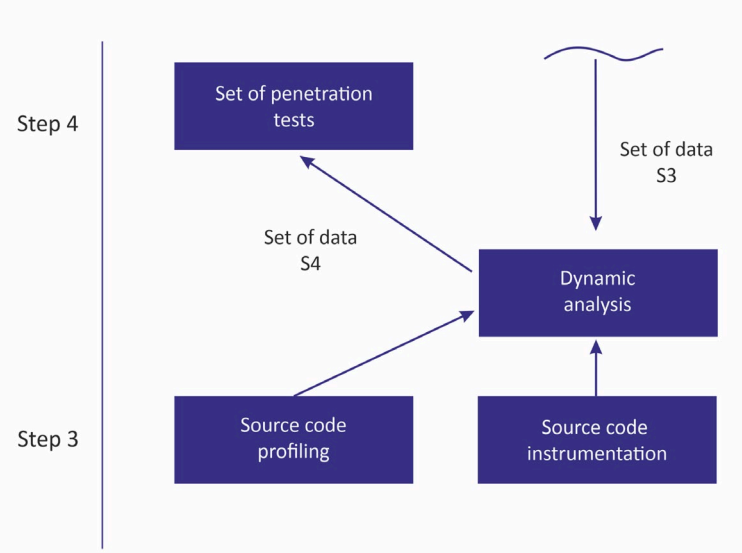

Figure 2: Stages 3 and 4 of the software vulnerability analysis methodology

Stage 5. Penetration testing.

Step 20. Analysis of the tests performed by the web application developers. Test optimization and improvement.

Step 21. Development of penetration tests based on the developed list of potential vulnerabilities and attack patterns.

Step 22. Installation of the test bench and performance of penetration tests using the developed tests. If new potential vulnerabilities are identified during the tests, new tests should added to the list of penetration tests. If it is found out during the test that it is necessary to specify the penetration test or extend it with any additional activities, the test should be corrected and repeated.

Step 23. Correction of the test set taking into account the changed set of input data. Repeated performance of corrected tests, as necessary.

Step 24. Determination of the relevant software vulnerabilities based on the penetration test results and preparation of reports.

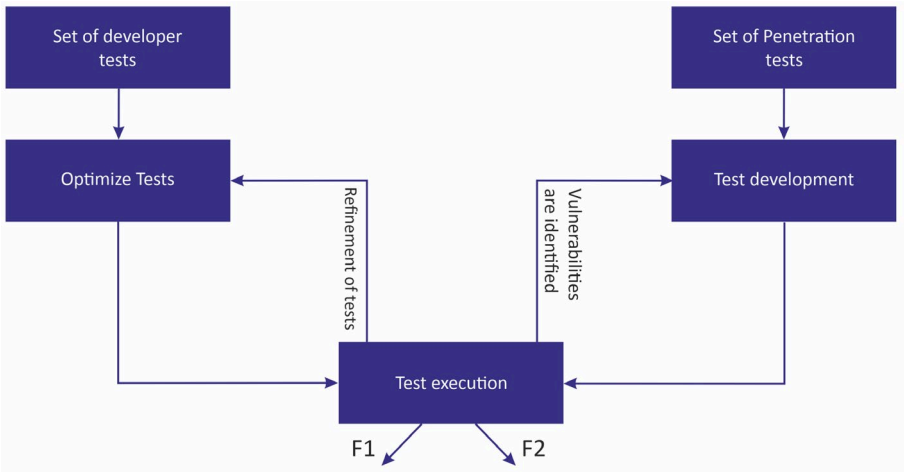

Figure 3: Stage 5 of the software vulnerability analysis methodology

3. Experimentation

Experimental research of the adapted procedure for software vulnerability analysis was performed by the experts of certified test laboratory in 2019-2021 using the research facilities of NPO Echelon.

The following objects were tested:

- The software subject to theme-based and certification tests in a certified test laboratory (group N1, 157 assessed objects);
- Open-source software (group N2, 91 assessed objects out of 157).

The experts of test laboratory performed the signature analysis of source texts using AppChecker (developed by NPO Echelon). The experts of test laboratory carried out the penetration tests using the recommendations provided by various thematic resources (CAPEC, OWASP) and the tool Scanner-VS (developed by NPO Echelon) [35]. The test benches used for penetration tests (step 7) were installed and set up by the experts of test laboratory in strict compliance with the requirements of operation and technical documents for the study objects.

4. Results of experimental studies

In the course of procedure approbation the experts of test laboratory identified 235 software vulnerabilities in the study objects of group N1. The relevance of all identified software vulnerabilities was confirmed by the software developer. Figure 4 shows the distribution of

identified vulnerabilities by types of successful attacks involving the identified vulnerability. A number of defects were found which can hardly be identified as intentional, though they can be used during cyber-attacks such as SQL code injection and incorrect operation of the access control mechanisms. The research has shown that the software includes explicit implants disguised as debugging tools such as embedded accounts and master passwords. Category Others includes less popular types of vulnerabilities such as XML injections or session fixation.

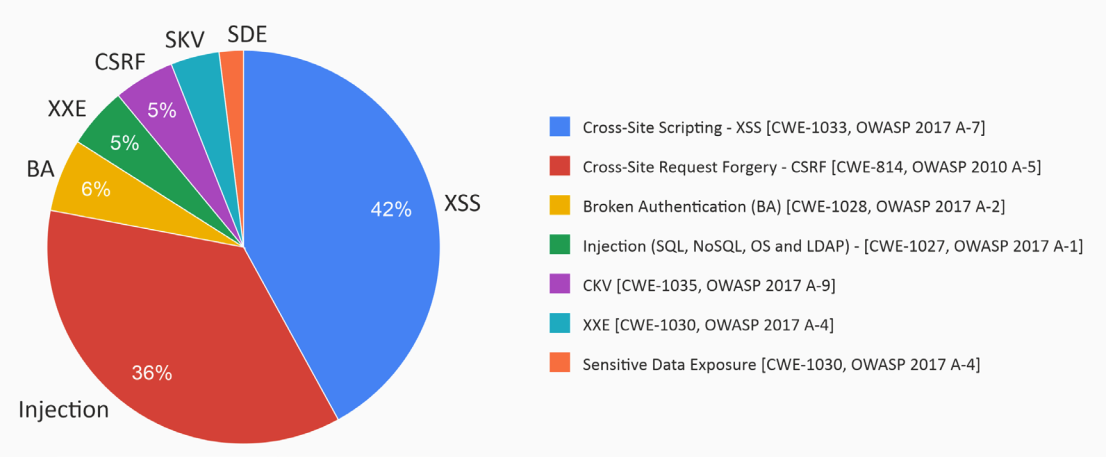

Figure 4: Distribution of identified vulnerabilities by types of Web application attacks

On average, it took the software developer one month to eliminate the vulnerability.

It should be noted that modern software complexes include open-source software modules. The research of group N2 (open-source software) has shown that such programs also include vulnerabilities. The research findings include 328 software defects (confirmed by the developers), of which 112 are the defects causing the software vulnerability. The software defects were found both using the static signature analysis of the software source code and by the dynamic analysis of the software code. The distribution of identified vulnerabilities is shown in Figure 5. The most popular types of identified software defects are the errors in DBMS queries (CWE-89, Improper Neutralization of Special Elements used in an SQL Command) and improper work with input data used for web page generation (CWE-79 Improper Neutralization of Input During Web Page Generation) (e.g. [36]).

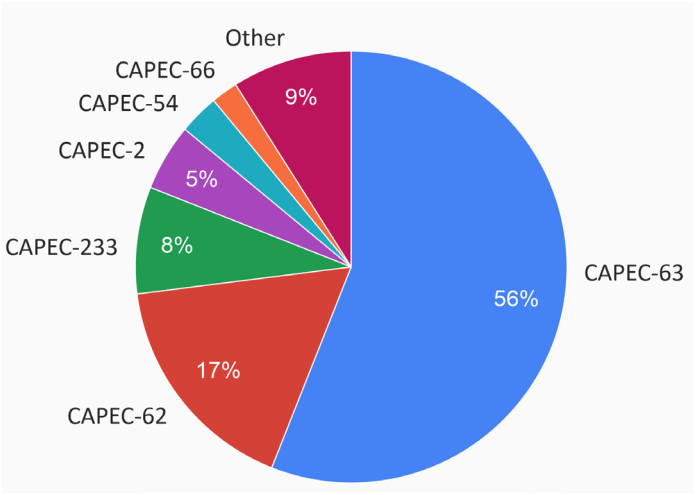

Figure 5: Software vulnerabilities distribution (open-source software) by the types of CWE defects

5. The problem state in foreign certification systems

It should be reminded that due to the innovations in foreign certification systems, reports of test laboratories, which provide an overview of the vulnerability analysis, are published on the official websites of the certification systems. The reports of test laboratories over the period of 2019-2021 were analyzed (sample group included 43 reports[3]) published on the website of NIAP, the regulating agency of the US certification system. The analyzed reports mostly included reports on the tests for compliance with the requirements of security profiles for network devices (28 reports). The rest of reports (5 reports) reflected the findings of tests for compliance with the security profile requirements for application software, operating systems, controls of access privileges policy and mobile device security tools.

The main findings of the completed analysis are given below.

1. In the course of all works, the test laboratories searched for the information about known vulnerabilities of the object to be certified in publicly available databases. Some test laboratories searched for known vulnerabilities not only by the key words directly relating to the object to be certified (the software name and version, software developer's name) but also by the identification data relating to borrowed components.

2. Only in half of the studies the test laboratories carried out additional penetration tests. Most works use the standard set of tests applicable to almost all types of certification objects working with web applications (e.g. network port scanning). Only one study included the information about penetration tests performed on the basis of potential vulnerabilities of the object to be certified formulated considering the analysis of the developer's evidence.

3. All studies related to the certification based on the security profile requirements for network devices included fuzzing tests. As a rule, automation software developed in-house was used in such cases. At the same time, the full-fledged dynamic analysis was not performed.

4. In their research, the test laboratories did not follow the guidelines of ISO/IEC TR 20004 pertaining to the development of the list of potential vulnerabilities based on CWE and CAPEC database analysis. This is due to the fact that the requirement to provide access to the source code of certified software is not mandatory in foreign certification systems. The analysis is carried out only within the scope compliant with the requirements of explanatory note; additional studies are performed only by a small number of test laboratories.

6. Conclusions

Based on the results of this study, it can be concluded that the combined software vulnerability analysis is effective and should be implemented in daily activities of experts in certified test laboratories. The analysis of web application vulnerabilities should be the first activity performed within the scope of certification tests, as part of the software analysis the developer carries out before marketing the product, and as part of the check for compliance with modern security standards because identification of vulnerabilities in the assessed object at later stages (e.g. after the start of certification tests or at the product support stage) implies the repetition of complete cycle of product tests and significant costs incurred by the developer. It should be noted that in case of certification tests it is recommended that known (confirmed) vulnerabilities of the object to be certified should be identified both at the initial and at the final stages of certification tests.

The following brief conclusions can be drawn based on the procedure approbation results:

- The number of identified vulnerabilities depends a lot on the processes of secure software development cycle existing in the software development company.
- The most critical vulnerabilities were found only if the access to software source code was provided.

[3] Reports on the products working with web applications or containing web applications were analyzed.

- The major part of vulnerabilities identified during the study could have been identified by the software developer at early stage of the software development using the methods of static and dynamic analysis of the software source code.

In order to reduce the number of vulnerabilities, it is recommended that web application developers should enhance the life cycle processes with the main activities aimed to develop secure software such as modeling of information security threats, static analysis of source texts, penetration tests. We believe that practical application of such procedures by Russian software developers will improve the security of created software and, consequently, will reduce significantly the number of information security incidents.

7. References

[1] P. M. Parker. The 2022 Report on Software Security Testing: World Market Segmentation by City. ICON Group International, 2021, 502 p.

[2] Petrenko S.A., Makoveichuk K.A., Olifirov A.V. Concept of cyber immunity of Industry 4.0. In: CEUR Workshop Proceedings. Selected Papers of the X Anniversary International Scientific and Technical Conference on Secure Information Technologies (BIT 2019). 2019. V. 2603. P. 93-99.

[3] Petrenko S. La Administraciyn de la Ciberseguridad. Industria 4.0. Oviedo, Asturias. University of Oviedo, 2019, 294 p.

[4] Petrenko A.A., Petrenko S.A., Makoveichuk K.A., Olifirov A.A. Methodological recommendations for the cyber risks management. In: CEUR Workshop Proceedings. 5. Ser. Selected Papers of the 5th International Scientific and Practical Conference Distance Learning Technologies - DLT 2020, 2021. V. 2914. P. 234-247.

[5] Korneev, N., Merkulov, V. Intellectual analysis and basic modeling of complex threats. CEUR Workshop Proceedings. 2019, vol. 2603, pp. 23-28.

[6] Dmitry P. Zegzhda, Igor Y. Zhukov. Aspects of information security of computer systems. In: CEUR Workshop Proceedings. Selected Papers of the XI Anniversary International Scientific and Technical Conference on Secure Information Technologies (BIT 2021). 2021.

[7] Demidov R.A., Pechenkin A.I., Zegzhda P.D. An approach to vulnerability searching of integer overflows in the executable program code. Automatic Control and Computer Sciences. 2018. V. 52. No 8. P. 1022-1028.

[8] A. Mazuera-Rozo, A. Mojica-Hanke, M. Linares-Vásquez and G. Bavota, "Shallow or Deep? An Empirical Study on Detecting Vulnerabilities using Deep Learning," 2021 IEEE/ACM 29th International Conference on Program Comprehension (ICPC), 2021, pp. 276-287, doi: 10.1109/ICPC52881.2021.00034.

[9] Y. Benslimane, Z. Yang and B. Bahli, "Information Security between Standards, Certifications and Technologies: An Empirical Study," 2016 International Conference on Information Science and Security (ICISS), 2016, pp. 1-5, doi: 10.1109/ICISSEC.2016.7885859.

[10] P. Stephanow and K. Khajehmoogahi, "Towards Continuous Security Certification of Software-as-a-Service Applications Using Web Application Testing Techniques," 2017 IEEE 31st International Conference on Advanced Information Networking and Applications (AINA), 2017, pp. 931-938, doi: 10.1109/AINA.2017.107.

[11] J. L. Hernandez-Ramos, S. N. Matheu and A. Skarmeta, "The Challenges of Software Cybersecurity Certification [Building Security In]," in IEEE Security & Privacy, vol. 19, no. 1, pp. 99-102, Jan.-Feb. 2021, doi: 10.1109/MSEC.2020.3037845.

[12] G. Ferreira, "Software Certification in Practice: How Are Standards Being Applied?," 2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C), 2017, pp. 100-102, doi: 10.1109/ICSE-C.2017.156.

[13] Barabanov A., Markov A. Modern Trends in the Regulatory Framework of the Information Security Compliance Assessment in Russia Based on Common Criteria. In Proceedings of the 8th International Conference on Security of Information and Networks (Sochi, Russian Federation, September 08-10, 2015). SIN '15. ACM New York, NY, USA, 2015, pp. 30-33. DOI: 10.1145/2799979.2799980.
[14] S. Dupont et al., "Incremental Common Criteria Certification Processes using DevSecOps Practices," 2021 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), 2021, pp. 12-23, doi: 10.1109/EuroSPW54576.2021.00009.
[15] M. Andrea, M. Philippe, D. Sbastien and G. Jeremy, "Towards Incremental Safety and Security Requirements Co-Certification," 2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), 2020, pp. 79-84, doi: 10.1109/EuroSPW51379.2020.00020.
[16] K. Li et al., "Tool support for secure programming by security testing," 2015 IEEE Eighth International Conference on Software Testing, Verification and Validation Workshops (ICSTW), 2015, pp. 1-4, doi: 10.1109/ICSTW.2015.7107462.
[17] C. Weir, I. Becker and L. Blair, "A Passion for Security: Intervening to Help Software Developers," 2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP), 2021, pp. 21-30, doi: 10.1109/ICSE-SEIP52600.2021.00011.
[18] Barabanov A., Grishin M., Markov A., Tsirlov V. Current Taxonomy of Information Security Threats in Software Development Life Cycle. In: 2018 IEEE 12th International Conference Application of Information and Communication Technologies (AICT). IEEE (17-19 Oct 2018, Almaty, Kazakhstan). 2018, pp. 356-361. DOI: 10.1109/ICAICT.2018.8747065.
[19] Barabanov A., Markov A., Tsirlov V. On Systematics of the Information Security of Software Supply Chains. Advances in Intelligent Systems and Computing. 2020. V. 1294. P. 115-129. DOI: 10.1007/978-3-030-63322-6_9.
[20] Barabanov A., Markov A., Fadin A., Tsirlov V., Shakhalov I. Synthesis of Secure Software Development Controls. In Proceedings of the 8th International Conference on Security of Information and Networks (Sochi, Russian Federation, September 08-10, 2015). SIN '15. ACM New York, NY, USA, 2015, pp. 93-97 DOI: 10.1145/2799979.2799998.
[21] R. Trifonov, O. Nakov, G. Pavlova, S. Manolov, G. Tsochev and P. Nakov, "Analysis of the Principles and Criteria for Secure Software Development," 2020 28th National Conference with International Participation (TELECOM), 2020, pp. 125-128, doi: 10.1109/TELECOM50385.2020.9299567.
[22] Taubenberger, S., Jürjens, J., Yu, Y. and Nuseibeh, B. (2013), "Resolving vulnerability identification errors using security requirements on business process models", Information Management & Computer Security, Vol. 21 No. 3, pp. 202-223. https://doi.org/10.1108/IMCS-09-2012-0054.
[23] H. Chen, D. Bao, H. Gao and J. Cheng, "A Security Evaluation and Certification Management Database Based on ISO/IEC Standards," 2016 12th International Conference on Computational Intelligence and Security (CIS), 2016, pp. 249-253, doi: 10.1109/CIS.2016.0064.
[24] D. Bao, W. Sun, Y. Goto and J. Cheng, "Development of Supporting Environment for IT System Security Evaluation Based on ISO/IEC 15408 and ISO/IEC 18045," 2018 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI), 2018, pp. 204-209, doi: 10.1109/SmartWorld.2018.00070.
[25] Varenitca V. V., Markov A. S., Savchenko V. V. Recommended Practices for the Analysis of Web Application Vulnerabilities. CEUR Workshop Proceedings. 2019. Volume 2603, pp. 75-78.

[26] V. Mounika, X. Yuan and K. Bandaru, "Analyzing CVE Database Using Unsupervised Topic Modelling," 2019 International Conference on Computational Science and Computational Intelligence (CSCI), 2019, pp. 72-77, doi: 10.1109/CSCI49370.2019.00019.
[27] V. Yosifova, A. Tasheva and R. Trifonov, "Predicting Vulnerability Type in Common Vulnerabilities and Exposures (CVE) Database with Machine Learning Classifiers," 2021 12th National Conference with International Participation (ELECTRONICA), 2021, pp. 1-6, doi: 10.1109/ELECTRONICA52725.2021.9513723.
[28] Varenitsa V., Markov A., Savchenko V., Tsirlov V. Practical Aspects of Vulnerability Detection During Certification Tests of Information Security Software. Voprosy kiberbezopasnosti [Cybersecurity issues]. 2021. No 5 (45), pp. 36-44. DOI: 10.21681/2311-3456-2021-5-36-44.
[29] Barabanov A., Markov A., Fadin A., and Tsirlov V. 2015. A Production Model System for Detecting Vulnerabilities in the Software Source Code. In Proceedings of the 8th International Conference on Security of Information and Networks (SIN '15). ACM, New York, NY, USA, 98-99. DOI: http://dx.doi.org/10.1145/2799979.2800019.
[30] Borzykh S., Markov A., Tsirlov V., Barabanov A. Detecting Code Security Breaches by Means of Dataflow Analysis. CEUR Workshop Proceedings, 2017. Vol. 2081. P. 15-20.
[31] Markov A.S., Fadin A.A., Tsirlov V.L. Multilevel Metamodel for Heuristic Search of Vulnerabilities in the Software Source Code, International Journal of Control Theory and Applications, 2016, vol. 9, No 30, pp. 313-320.
[32] F.M.Tudela and etc. On Combining Static, Dynamic and Interactive Analysis Security Testing Tools to Improve OWASP Top Ten Security Vulnerability Detection in Web Applications. Appl. Sci. 2020, 10(24), 9119; https://doi.org/10.3390/app10249119
[33] Barabanov A.V., Markov A.S., Tsirlov V.L. Information Security Controls Against Cross-Site Request Forgery Attacks On Software Application of Automated Systems. Journal of Physics: Conference Series. 2018. V. 1015. P. 042034. DOI :10.1088/1742-6596/1015/4/042034
[34] Reber, G., Malmquist, K., Shcherbakov, A. 2014. Mapping the Application Security Terrain. Voprosy kiberbezopasnosti [Cybersecurity issues]. 2014. N 1(2). P. 36-39. DOI: 10.21681/2311-3456-2014-2-36-39.
[35] Markov A., Fadin A., Shvets V., Tsirlov V. The experience of comparison of static security code analyzers. International Journal of Advanced Studies. 2015. V. 5. No 3. P. 55-63. DOI: 10.12731/2227-930x-2015-3-9
[36] D. Gonzalez, F. Alhenaki and M. Mirakhorli, "Architectural Security Weaknesses in Industrial Control Systems (ICS) an Empirical Study Based on Disclosed Software Vulnerabilities," 2019 IEEE International Conference on Software Architecture (ICSA), 2019, pp. 31-40, doi: 10.1109/ICSA.2019.00012.