ARTICLE

UDC 004.4

MODERN METHODS OF AUTOMATED SOFTWARE SECURITY ANALYSIS: FROM STATIC ANALYSIS TO COMPREHENSIVE APPROACH

Volkova A.A.

National Research University of Electronic Technology, Moscow,
e-mail: alna_volkova_02@inbox.ru

The article presents a comprehensive analysis of modern methods for automated software security analysis. The main approaches to vulnerability detection are considered, including static and dynamic code analysis, as well as hybrid methods using machine learning. Current problems and limitations of existing tools are analyzed, and promising directions for the development of automated security testing technologies are proposed. The study covers a wide range of issues, including the effectiveness of various analysis methods, problems of integrating tools into the development process, and the possibilities of using artificial intelligence to improve the accuracy of vulnerability detection. Special attention is paid to the problem of reducing false positives and prioritizing identified vulnerabilities. Based on the analysis, recommendations are formulated for improving existing approaches and developing new methods for automated software security analysis. The research highlights the critical importance of developing comprehensive approaches that combine different analysis methods and leverage modern machine learning technologies to enhance the effectiveness of vulnerability detection. The results show that no single method can provide full coverage of all types of vulnerabilities, necessitating the use of a hybrid approach integrating static and dynamic analysis with advanced AI techniques. The article also addresses issues of standardizing vulnerability descriptions, challenges in creating high-quality datasets for training machine learning models, and prospects for developing security analysis tools in the context of modern software development methodologies such as DevOps and continuous integration. The study emphasizes the need for ongoing research and development in this field to keep pace with evolving security threats and increasingly complex software systems. It also discusses the importance of balancing automation with human expertise in the security analysis process and the potential for AI-driven tools to augment rather than replace human security analysts.

Keywords: static analysis, dynamic analysis, software vulnerabilities, automated testing, software security, machine learning, security testing

Introduction

In the modern world, software plays a critical role in virtually all spheres of life – from household devices to industrial control systems, making security issues particularly relevant. The growing complexity of software systems and a significant increase in source code volumes, reaching millions of lines, make complete manual vulnerability testing practically impossible. In this regard, automated security analysis tools are becoming a necessary element of the software development process, allowing potential problems to be identified at early stages [1-3].

Existing solutions for automated security analysis include various approaches and methods that work at different stages of the software lifecycle. Static code analysis allows finding vulnerabilities without executing it, dynamic analysis checks security during program execution, and hybrid methods combine both approaches. However, their effective application faces a number of significant limitations, including a high level of false positives, reaching 30-50% of the total number of problems found, the complexity of configuration and integration into the development process, as well as insufficient coverage of various types of vulnerabilities [4].

An important problem is also the standardization of vulnerability descriptions – different tools use their own classifications and terminology, which makes it difficult to compare them and integrate analysis results. Existing initiatives such as Common Weakness Enumeration (CWE) and Common Vulnerabilities and Exposures (CVE) aim to create a unified classification system, but the process of mapping rules from different tools to these standards often leads to inconsistencies and gaps [5].

Special attention should be paid to the problem of "false negative" results, when existing vulnerabilities are not detected by analysis tools. Unlike false positives, it is much more difficult to assess the scale of this problem due to its very nature. At the same time, missed critical vulnerabilities can pose a serious threat to software security.

A promising direction for the development of security analysis tools is the application of machine learning methods, which allow increasing the accuracy of vulnerability detection and reducing the number of false positives.

However, the effective use of such approaches requires high-quality datasets containing examples of real vulnerabilities and their exploitation. Creating and maintaining the relevance of such datasets presents a separate challenging task.

Comprehensive analysis of the effectiveness of methods and tools for automated software security testing

As part of this study, a comprehensive analysis of existing methods and tools for automated software security analysis was conducted. The research was based on a systematic review of scientific literature, analysis of practical experience in applying various tools, as well as an assessment of promising directions for technology development in this area. To form a representative sample, more than 160 thousand vulnerability records from the Common Vulnerabilities and Exposures (CVE) database were analyzed, covering both proprietary and open-source projects.

The methodology included several interrelated stages. At the first stage, an analysis of existing approaches to vulnerability classification was conducted, including Common Weakness Enumeration (CWE) and specialized catalogs of major vendors. Significant discrepancies in terminology and classification between different systems were identified, which creates difficulties in integrating analysis tools [6, 7].

At the second stage, the features of applying static and dynamic code analysis were investigated. It was found that static analyzers, despite a high percentage of false positives (up to 30-50%), allow identifying potential vulnerabilities at early stages of development. Dynamic analysis demonstrates higher accuracy but requires the creation of special test scenarios.

Special attention was paid to studying the following key aspects:

• The effectiveness of various approaches to vulnerability detection, including code coverage analysis, detection accuracy, and false positive rate.

• Problems of integrating analysis tools into the development process, in particular issues of configuration, performance, and usability.

• Possibilities of applying machine learning to improve analysis accuracy, including the use of large datasets of known vulnerabilities.

• Methods for reducing the number of false positives by improving contextual analysis and prioritizing results.

• Approaches to prioritizing identified vulnerabilities based on criticality assessment and probability of exploitation.

The study also included an analysis of practical experience in using security tools in real projects. The main reasons for refusing to use analysis tools were identified, including configuration complexity, high cost, insufficient integration with development processes, and low quality of problem descriptions.

To improve the effectiveness of security analysis tools, it is necessary to:

• Improve integration with modern development environments.

• Provide more accurate prioritization of identified vulnerabilities.

• Provide more detailed problem descriptions and recommendations for their elimination.

• Reduce the number of false positives by improving contextual analysis.

• Expand support for various programming languages and technologies.

Additionally, issues of standardizing vulnerability descriptions and problems of mapping different classifications were considered. The analysis showed that existing initiatives such as Common Weakness Enumeration (CWE) and Common Vulnerabilities and Exposures (CVE), while providing a basic structure for classification, do not always allow accurate description of complex attack scenarios and relationships between various vulnerabilities [8].

Special attention was paid to the problem of "false negative" results, when existing vulnerabilities are not detected by analysis tools. Unlike false positives, it is much more difficult to assess the scale of this problem. Potential approaches to assessing and reducing the number of missed vulnerabilities were considered, including the application of machine learning methods and combining different types of analysis.

The study also touched upon issues of creating and maintaining up-to-date datasets for training machine learning models in security analysis tasks. Existing open vulnerability databases and their limitations were analyzed, and approaches to creating synthetic datasets for testing analysis tools were considered.

As a result of the analysis, recommendations were formulated for improving methods and tools for automated software security analysis, including the need to develop more flexible formats for describing analysis results, improving integration with modern development processes, and applying machine learning methods to increase the accuracy of vulnerability detection.

Comparative analysis of modern automated vulnerability search tools and prospects for their development using machine learning methods

The analysis showed that existing automated security analysis tools can be divided into several main categories:

1. Static code analysis tools that work without executing it, including syntactic analyzers, semantic data flow analyzers, and model checking tools. These solutions allow finding potential vulnerabilities at early stages but have limitations in the form of a large number of false positives.

2. Dynamic analysis tools that check security during program execution by monitoring application behavior, tracking data flows, and identifying anomalous behavior. Such tools provide higher accuracy but require careful preparation of test scenarios.

3. Hybrid solutions combining various analysis approaches, including static and dynamic analysis, as well as machine learning methods. Such tools allow compensating for the shortcomings of individual approaches and increasing the effectiveness of vulnerability detection.

4. Specialized tools for analyzing specific types of vulnerabilities, such as SQL injections, cross-site scripting (XSS), or buffer overflow. Such solutions provide in-depth analysis of certain vulnerability classes.

However, each approach has its advantages and limitations. Static analysis is effective for early detection of potential security problems but often generates a significant number of false positives, which complicates practical application. According to research, up to 50% of static analyzer warnings can be false. Dynamic analysis provides higher accuracy by checking the actual behavior of the program but requires creating special test scenarios and cannot detect all potential problems due to limited code coverage [9].

The study showed that a promising direction is the application of machine learning methods to improve analysis accuracy and reduce the number of false positives. The use of neural networks and other machine learning algorithms allows considering the context of software use, identifying complex vulnerability patterns, and adapting analysis to specific requirements. For example, applying deep learning methods for source code analysis can reduce the number of false positives by 30-40% compared to traditional approaches [10, 11].

An important aspect is also the integration of security analysis tools into the software development process. Research shows that the effectiveness of tool application significantly depends on how conveniently they fit into existing development processes and tools. It is necessary to ensure support for popular development environments, version control systems, and continuous integration tools.

Special attention should be paid to the problem of prioritizing identified vulnerabilities. With limited resources, it is critical to correctly determine the sequence of addressing detected security problems. A promising approach is the application of risk-oriented analysis, considering both the probability of vulnerability exploitation and the potential damage from its use.

Additionally, the following aspects of developing automated security analysis tools were considered:

- Application of natural language processing methods for analyzing comments in code, documentation, and other textual artifacts of the project to identify potential vulnerabilities.
- Use of graph neural networks to analyze code structure and identify complex dependencies between various programs components.
- Development of interpretable machine learning methods to ensure transparency and explainability of analysis results.
- Creation of specialized language models pre-trained on large volumes of source code to improve analysis accuracy in specific subject areas.
- Application of active learning methods to adapt machine learning models to the specifics of projects and reduce the need for large volumes of labeled data.
- Development of analysis methods that consider the features of modern software architectures, including microservices and distributed systems.

The study also touched upon issues of evaluating the effectiveness of security analysis tools. Various metrics used to compare tools were considered, including accuracy, completeness, F1-measure, and analysis time. The need to develop more comprehensive evaluation methods that consider not only technical aspects but also usability, integration with development processes, and economic efficiency was noted.

Conclusion

The conducted study confirms the critical importance of developing methods for automated software security analysis. The results show that the most promising approach is a

comprehensive one, combining various analysis methods and using modern machine learning technologies to improve the effectiveness of vulnerability detection. Analysis of existing tools revealed that no single method can provide complete coverage of all types of vulnerabilities, making it necessary to apply a hybrid approach.

Static code analysis, despite a high level of false positives, remains an important component of the security testing process, allowing potential problems to be identified at early stages of development. Dynamic analysis, in turn, provides more accurate results by analyzing the actual behavior of the program, but requires significant resources to create test scenarios. Integration of machine learning methods can significantly improve analysis accuracy by considering the context of software use and adapting to specific project requirements.

Further research in this area should be directed towards developing more advanced methods of integrating analysis tools into the development process. It is critically important to reduce the number of false positives, which significantly reduce developers' trust in the results of automated analysis. It is also necessary to pay attention to the problem of false negative results, which can miss critical vulnerabilities.

Special attention should be paid to developing methods for prioritizing identified vulnerabilities and automating the process of their elimination. Existing approaches to vulnerability ranking are often based on simplified criticality assessment models that do not consider the specifics of projects. It is necessary to develop more advanced risk assessment methods that consider both technical aspects of vulnerabilities and features of business processes.

Integration of security analysis tools into modern DevOps processes presents a separate important task. It is necessary to ensure continuous security analysis at all stages of the software lifecycle without creating significant delays in the development process. A promising direction is the development of intelligent systems capable of automatically determining the optimal set of security checks depending on the context of code changes.

Standardization and unification of security analysis results presentation also requires further development. Existing standards such as CWE and CVE provide a basic classification of vulnerabilities, but do not always allow accurate description of complex attack scenarios and relationships between various vulnerabilities. It is necessary to develop more flexible formats for describing analysis results, ensuring effective communication between various participants in the development process.

References

1. Terentyeva Yu.Yu. Modeling of communication systems in terms of ensuring its stability // Devices and systems. Management, control, diagnostics. 2024. Is. 2. P. 64-70. DOI: 10.25791/pribor.2.2024.1479.

2. Birikh E.V., Gruzdev A.S., Kamalova A.O., Sakharov D.V. The choice of tools for dynamic security analysis of web applications for the tasks of the digital economy // Information protection. Insider. 2024. Is. 1(115). P. 42-46.

3. Lapina M.A., Aganesov A.S., Koronsky A.A., Khodakov M.I. Comparative characteristics of software code analyzers // Student science for the development of information society: Materials of the XU All-Russian Scientific and Technical Conference with the invitation of foreign scientists, Stavropol, November 28, 2023. Stavropol: North Caucasus Federal University, 2024 P. 425-431.

4. Bukarev A.V. Effective method of automated software testing of consumer electronics devices using cloud devices // Engineering Bulletin of the Don. 2023. Is. 9(105). P. 212-219.

5. Bukarev A.V. Analysis of statistical characteristics of the process of automated testing of mobile applications using automated process control systems // Prospects of science. 2023. Is. 2(161). P. 39-42.

6. Gimatdinov D.M., Gerasimov A.Y., Privalov P.A. et al. An Automated Framework for Testing Source Code Static Analysis Tools // Proceedings of the Institute for System Programming of the RAS. 2021. Vol. 33, Is. 3. P. 41-50. DOI: 10.15514/ISPRAS-2021-33(3)-3.

7. Motorin S.V., Kalyakina D.P., Motorin A.S. Analysis of the impact of test automation on the example of the coronapay mobile application // Research Forum – 2024: collection of articles of the International Scientific and Practical Conference, Petrozavodsk, January 09, 2024. Petrozavodsk: International Center for Scientific Partnership "New Science" " (IP Ivanovskaya I.I.), 2024. P.165-173.

8. Sobolevsky V.A. Using AUTOML technologies to solve monitoring problems // Informatization and Communications. 2024. Is. 1. P. 90-97. DOI: 10.34219/2078-8320-2024-15-90-97.

9. Maksimova E.A., Danilin E.D. Software development of the information and analytical module of the process monitoring system at the user's automated workplace // Student science for the development of the information society: Materials of the XU All-Russian Scientific and Technical Conference with the invitation of foreign scientists (Stavropol, November 28, 2023). Stavropol: North Caucasus Federal University, 2024. P. 187-194.

10. Gulyaev D.A., Gulyaeva A.V. Software security: challenges and innovations // Planning, conducting and interpreting the results of scientific research: Collection of articles of the International Scientific and Practical Conference (Kirov, January 20, 2024). Ufa: Aeterna LLC, 2024. P. 43-45.

11. Dvoryak D.A. The influence of the choice of a software platform on the security of web applications // Young Scientist. 2024. Is. 7(506). P. 7-10.