Eng. Cov & Innov. An Int. J. Volume 01, Issue 02, May – August, 2026, ISSN 3108-1843 (online)

Universal Research Forum
Engineering Convergence and Innovation (ECI) An International Journal
www.universalresearchforum.com

An Intelligent Software Threat Monitoring Platform for Continuous Vulnerability and Malware Detection

Shashikala Dasara[1,*], Rutuja Pakhare[2], Sanjana Zangaruche[2], Pallavi Kundekar[2], Sairaj Desai[2]

[1]Assistant Professor of Computer Science and Engineering (AIML) Jain College of Engineering and Research , Udyambag, Belagavi, India
[2]Department of Computer Science and Engineering (AIML) Jain College of Engineering and Research ,Udyambag, Belagavi, India
*Email: shashikalareddy1120@gmail.com , rutujapakhare23@gmail.com , sanjana.zangaruche05@gmail.com , pallavikundekar6@gmail.com , desaisairaj21@gmail.com & Mob: 6363043725, 9380943219.

Published Date: 04 June 2026.

ABSTRACT

Today's software development is highly dependent on automated CI/CD pipeline solutions that help deliver applications rapidly. Nonetheless, the security mechanisms provided by CI/CD pipelines are far from optimal and fail to provide a holistic approach to security. They are not able to provide any protection from known or unknown vulnerabilities or malware hidden in compiled binaries and software supply chain components. The traditional approaches to security like SAST, SCA and DAST can be deployed separately and aim to detect only known vulnerabilities without any capacity to identify unknown and zero-day attacks at the binary level. The current paper provides a comprehensive literature review and identifies critical issues such as fragmentation of toolchains, lack of interoperability, and absence of AI-powered systems among others. As a solution to these problems, this paper introduces Zentronyx - An Intelligent Software Threat Monitoring Platform for Continuous Vulnerability & Malware Detection ,capable of identifying vulnerabilities and malware continuously throughout the software development process. Zentronyx is built based on SAST, SCA and DAST and is supplemented by a GNN-based malware detector the core innovation provided by this platform. Specifically, this GNN component performs analysis of the compiled binary objects by representing the binary objects in a graph form (including control flow graphs and function call graphs) and applying the structure learning abilities of GNNs to identify novel, zero-day threats, which cannot be detected by signature-based and flat ML techniques. In addition, the platform uses policy-driven decision functionality as well as a centralized visualization dashboard, which provides capabilities for real-time risk assessment, explainability of GNN findings, and automated decisioning to allow or block suspicious binaries. Contributions made by this paper can be highlighted as follows: an identification of limitations in existing frameworks, a creation of a unified continuous monitoring framework, and a combination of GNN analysis capabilities directly within automated CI/CD pipelines.

Keywords - *DevSecOps, CI/CD security, software supply chain security, SAST, SCA, DAST, GNN-based malware detection, binary malware detection.*

1. INTRODUCTION

CI/CD pipeline integration, testing, and delivery of code are popular in contemporary software development [11]. The employment of CI/CD pipeline assists developers in delivering applications faster, performing fewer actions, and obtaining feedback throughout the development period [9]. Hence, companies become more efficient and flexible owing to

their ability to deliver software rapidly [10]. However, the CI/CD pipeline may pose several security risks as well. Some of the security practices may be absent when implementing the CI/CD pipeline, meaning that testing, for example, might occur separately in the forms of code, dependency, and runtime tests [4]. Besides, using third-party software raises the likelihood of encountering some vulnerabilities which will be difficult to detect [14]. The purpose of the suggested solution, called Zentronyx, is addressing these problems.

While it is true that modern CI/CD solutions offer capabilities for the fast development of software applications, the majority of them lack the existence of a security tool that could be leveraged in the process of looking for software weaknesses as well as malware. The processes of static application testing, software composition analysis, and dynamic application testing are conducted separately and primarily help detect already known vulnerabilities, but nothing is done about detecting any malicious activity in the compiled code and in third-party components . Moreover, there are no automation possibilities at the moment, and all security actions are being taken manually.

1.1. System Background and Related Work

i) CI/CD Pipeline Overview

A CI/CD pipeline refers to an automated mechanism that enables continuous integration, continuous testing, and continuous deployment of applications [11]. In this regard, the pipeline is initiated by programmers' committing codes in the same repository for a series of processes such as building, testing, and deploying. For instance, during the build process, the coding is converted to executable, whereas the tests are conducted at the following step to confirm correctness and robustness [10]. The validated software is later deployed in a staging or production environment for efficient deployment of the software. However, despite providing efficiency in the software development process, CI/CD pipelines lack effective implementation of security measures [4]. In most cases, security mechanisms such as static code analysis, supply chain analysis, and dynamic application testing are conducted independently, leading to loopholes within the security analysis process. Additionally, current CI/CD pipelines lack the incorporation of security mechanisms that check compiled binaries and third-party libraries [14].

ii) Static Application Security Testing (SAST)

Static Application Security Testing (SAST) can be considered as a solution that provides for the examination of source code in order to discover potential flaws in its structure [7]. Such testing allows finding coding errors, improper programming techniques and vulnerabilities, and therefore can be regarded as helpful in the early detection of security flaws. Static code analysis conducted during the software development process via SAST allows applying shift-left testing techniques and detecting security issues prior to building the application [4]. By analysing the source code of the program, the system uncovers vulnerabilities, such as SQL injection, cross-site scripting, and input validation [7]. In line with its very name, SAST testing is performed at the early stage of the source code cycle life without actually executing any code [7]. Consequently, such a testing technique enables identifying vulnerabilities early on and minimizing the cost of dealing with security issues. That is why the use of SAST is crucial for creating secure CI/CD pipelines. Nonetheless, some drawbacks should also be taken into account; for example, this type of testing deals with the source code only and may lead to false positives.

iii) Dynamic Application Security Testing (DAST)

Dynamic Application Security Testing (DAST), also called interactive application security testing, evaluates the security of an application during runtime and without using the source code of the application. This process involves communicating with the application and evaluating the security of the application as accurately as possible to discover potential flaws such as insecure authentication, injections, and misconfigurations [10]. DAST helps identify security flaws that occur during the runtime and that could not have been discovered during the coding process since it evaluates the application once it is already implemented [4]. However, this testing method is incapable of identifying the cause of vulnerabilities since the code is inaccessible [11].

iv) Software Composition Analysis (SCA)

Software Composition Analysis is one of the techniques employed to uncover vulnerabilities related to third-party components used in software applications. Contemporary software solutions depend heavily on a variety of third-party components, and the inclusion of vulnerable third-party components in the software presents a great danger to the software solution. Using Software Composition Analysis technique,

programmers can check their software components against existing vulnerability databases to establish whether they are dealing with vulnerable third-party components or not. It also helps comply with software license requirements [4]. Nevertheless, it cannot detect zero-day vulnerabilities [9].

v) Binary and Malware Analysis

Malware detection and binary analysis includes the process of examining the compiled code of the software in order to find out if there is any kind of code embedded within that is harmful and could not be found through any other method. This is required because sometimes a threat is inserted into the software during the process of development or due to vulnerabilities in dependencies [13]. The different kinds of methods that can be used for this purpose are signature based, behavioral detection, and machine learning algorithms [8].

1.2 Literature Review

Today, these methods remain disjointed, in which case vulnerability detection, pipeline security, and malware analysis are considered separately. While DevSecOps promotes the early adoption of security tools, the process is non-standard and non-scalable. While AI-powered approaches offer improved automation capabilities, they suffer from shortcomings related to integration and validation, and while GNN-based methods are highly accurate, they involve significant computational overheads. Furthermore, there is no smart binary-level threat detection within current CI/CD frameworks.

i) Review of Existing Approaches

Current research efforts on software security are predominantly centered around software security tools such as SAST, SCA, and DAST within CI/CD pipelines. The former tool detects software vulnerabilities through source code analysis, but it is only capable of detecting known vulnerability patterns and can return false positives. In turn, SCA detects any software vulnerability through third-party dependencies that can be found within an already-known database. Still, the approach does not allow discovering newly-created or deliberately camouflaged malware. Finally, the last approach detects vulnerabilities within running applications and allows discovering behavioral vulnerabilities, but this method requires deploying an application first and takes considerable time. Recently, researchers considered developing malware detection models with the use of machine learning techniques based on various features, such as opcodes and entropy. The advanced technique is called Graph Neural Network (GNN) and aims to increase detection accuracy by studying binary structural patterns. Nonetheless, such an approach poses problems associated with the computational intensity of the task and difficulty of implementing the solution in a timely manner. Finally, DevSecOps frameworks aim to introduce cybersecurity into CI/CD pipelines, but their implementation currently lacks centralized intelligence

1.3 Research Gaps

From the above literature survey, we can see that some significant issues remain in the current approaches. First, there is no comprehensive DevSecOps approach to integrate security in an inconsistent manner across the entire procedure. Second, the AI-driven solution faces issues related to scalability, integration, and validation. Third, there is no provision in CI/CD tools for threat detection at higher levels and for binary analysis, which makes it incapable of detecting advanced threats and unknown malware attacks. Fourth, the computation cost is quite high when using the GNN approach. Fifth, the benchmark datasets are not available.

Objectives

- For the inclusion of Static Application Security Testing (SAST), Software Composition Analysis (SCA), and Dynamic Application Security Testing (DAST) within a single security tool.
- For the development of a Graph Neural Network-based malware detection system in compiled binaries.
- For the detection of novel attacks using behavior and structural analysis techniques.
- For the creation of a CI/CD security pipeline that would perform automatic security analysis.
- For the production of a comprehensive security report regarding the results of vulnerability detection, dependency risks, and malware detection.

2. METHODOLOGY

2.1 System Overview

The Zentronyx intelligent software platform is designed to provide continuous scanning and detection of vulnerabilities and malware in automated CI/CD

pipelines. The platform uses a blend of both traditional security analysis techniques and machine learning to resolve the problem of fragmentation found in contemporary DevSecOps platforms [9], [3]. Being the core strength of the platform, Zentronyx provides an integration between Static Application Security Testing (SAST), Software Composition Analysis (SCA), and Dynamic Application Security Testing (DAST), as well as Graph Neural Network (GNN)-based malware detection technique that forms the basis of the platform. While using traditional methods to find vulnerabilities in the source code, dependencies, and run time, GNN allows the framework to scan vulnerabilities within compiled binaries to uncover hidden malware [14].

2.2 Architecture Design

The architecture followed by Zentronix allows the implementation of a modular approach with a pipeline that can be easily combined with CI/CD tools such as GitHub Actions, GitLab CI, and Jenkins. The architecture consists of a number of phases of security conducted sequentially as well as in parallel fashion. The pipeline begins with code integration and goes through the standard phases of security analysis such as SAST, SCA, and Build. After the binaries are created, the malware detection phase through GNN becomes an essential part of the intelligent module within the process flow. In this step, binaries are transformed into graphs and analyzed through GNNs.

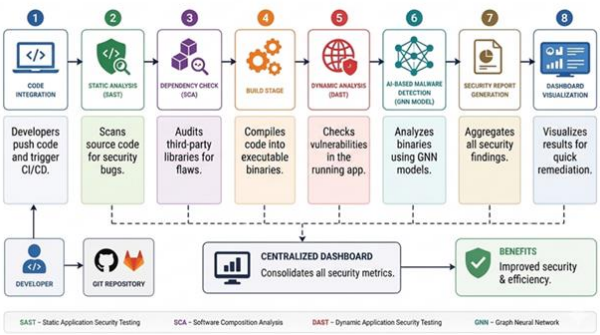

Fig. 1 Zentronyx: Unified DevSecOps Workflow with AI-Driven Malware Detection

2.3 Workflow Description

The process flow of Zentronyx starts off with several validation processes carried out systematically through code commits through the CI/CD process. The whole process begins from the moment developers submit their changes to their code in a version control system; hence beginning the CI/CD process but improved by using Zentronyx. At this point, there will be an SAST process where the source code is analyzed for any coding weakness, flaws, and any other security risks before the software application program runs. Secondly, the process continues to SCAs where third-party libraries and dependencies are reviewed for any possible threats and outdated source codes.

a.Code Integration :

Developers push code changes to a version control system, automatically triggering the Zentronyx-enhanced pipeline.

b. SAST Analysis :

Source code undergoes Static Application Security Testing (SAST) to identify coding vulnerabilities, logic flaws, and insecure practices without execution.

c. SCA Analysis :

Software Composition Analysis (SCA) evaluates third-party libraries and dependencies for known vulnerabilities, outdated components, and supply-chain risks.

d. Build and Binary Generation :

The application is compiled into executable binary artifacts.

e. Malware Detection (GNN-based Module) :

This generated binary is analyzed through the malware detection system, Zentronyx's main technological novelty, which leverages a Graph Neural Network (GNN). First, the binary is decompiled to generate control-flow graphs (CFGs) or function call graphs. These graphs are used as input for training the GNN model, which successfully recognizes the relationship and behavior of multiple functions or API calls in the software code. Through this process, the GNN algorithm assigns a malware probability score, detecting zero-day attacks that cannot be identified using conventional techniques.

f. DAST Testing

Here, the software is made to run in an environment that ensures DAST can be carried out. Static testing is when the software is examined when it is at rest; dynamic testing is when the way in which the software works is examined, allowing the tester to find any

vulnerability created by the running of the software. Poor authentication, input validation, and behavior of the software are examples of some of the issues that can be found using DAST.

g. Reporting and Decision

The findings gathered at each level will be integrated into a comprehensive security report, complete with risk scores. Based on the findings, the policy engine will make its decision whether to approve or disapprove deployment. The centralized dashboard provides real-time monitoring features, along with GNN explanations and malware score.

2.4 Advantages of Proposed System

The solution provided by Zentronyx holds many benefits compared to legacy and fragmented approaches. Firstly, the integration of SAST, SCA, DAST, and a malware detection component based on the application of GNNs creates a continuous pipeline that ensures end-to-end threat protection along with overcoming the drawbacks of fragmented systems [9], [3]. The unique feature of the system is the utilization of the GNN component that provides excellent capabilities for recognizing malware by analyzing the structure of the graph representing the binary code [8], [13]. The result is better threat detection and decreased remediation costs. Secondly, the policy engine allows enforcing security guidelines, and the centralized dashboard ensures visibility and explainable results.

3. COMPARATIVE ANALYSIS

3.1 Comparison of Techniques

However, each one of the security technologies has certain strengths when using the CI/CD process, although not any of them have significant performance levels on their own [3][9]. Some of the most popular methods that have been used within the industry are static application security testing (SAST), software composition analysis (SCA), dynamic application security testing (DAST), and malware detection using graph neural networks. First, SAST technology is commonly used for detecting bugs in the initial stage because it conducts analysis without actually executing the application. The method may discover issues concerning programming and coding [7]. Second, SCA is crucial for identifying vulnerabilities in the third-party components and packages which may affect the security of the entire software package. SCA solves the issue of software supply chain risk that emerged due to the increasing number of open-source software solutions [14]. Third, DAST examines whether the application is working properly by testing its response to real attacks. Thus, DAST is capable of detecting runtime vulnerabilities like incorrect authentication and misconfigurations [10]. Finally, malware detection through the application of GNN makes use of graphs and their algorithms for determining malware in binary files. Consequently, the approach can detect unknown and more complex forms of malware [8][15]. While each of the technologies has its unique benefits, using each.

Table 1 Different techniques of security testing

Technique	Key Capabilities	Contribution to Zentronyx
SAST	Early detection of code-level vulnerabilities; supports secure coding practices	Enables early-stage security validation and reduces coding errors
SCA	Identifies risks in third-party dependencies and open-source components	Strengthens software supply chain security
DAST	Detects runtime vulnerabilities and configuration issues	Ensures secure application behavior during execution
GNN-based Malware Detection	Captures complex relationships in binaries using graph structures; detects unknown and zero-day threats	Enhances intelligent and explainable malware detection in binary artifacts
Integrated Approach (Zentronyx)	Combines multi-layer security with automation and intelligence	Provides end-to-end, comprehensive pipeline security

3.2 Performance Evaluation

The suggested system referred to as Zentronyx, ensures more efficient security performance due to implementation of various security measures in one CI/CD pipeline [6]. Unlike in case of existing solutions in which SAST, SCA, and DAST are employed independently of each other, in Zentronyx several types of analysis are performed at different SDLC phases [11]. In such a way, it becomes possible to find

vulnerabilities from such perspectives as source code, dependency, run-time, and binary, contributing to more efficient detection and higher coverage rate [5]. The most important aspect related to this system is the usage of GNN-based malware detection allowing discovering complex malware based on structural connection in binaries [12][13]. Moreover, automation ensures easy identification and constant protection of the application throughout its lifecycle [2]. As a result, the following advantages of the considered solution can be mentioned as higher coverage and improved visibility through centralized reporting, quick reaction time, and protection at all SDLC phases.

i). Detection Effectiveness

Zentronix increases the efficiency of the detection process through the incorporation of Static Application Security Testing (SAST), Software Composition Analysis (SCA), Dynamic Application Security Testing (DAST), and binary analysis in one pipeline [6]. The combination of these methods allows for the detection of vulnerabilities at various stages such as source code, dependencies, runtime, and binary files [10].

ii). ML-Based Malware Detection Performance

The system utilizes a malware detection model with the assistance of machine learning and uses a Graph Neural Network (GNN) [1]. The model is capable of analyzing various properties of binary files, including entropy, opcode sequences, and control flow dependencies [15]. The model will be able to output a probability score that determines whether a file is a malware program [8].

iii). Pipeline Efficiency and Automation

Zentronyx makes the process more efficient through the automation of security tests across all stages of CI/CD pipelines [2]. Continuous testing at every stage eliminates manual labor while facilitating vulnerability detection [3]. In addition, Zentronyx offers rapid feedback to the developers for timely corrections without compromising security compliance [6].

4. CHALLENGES AND FUTURE WORK

4.1 Challenges

The integration of a unified security platform such as Zentronyx into a CI/CD pipeline is faced with some hurdles with regard to guaranteeing effective integration of the analysis processes without causing a delay in the CI/CD pipeline. This is in addition to making sure that all modules have similar levels of threat detection.

i). Integration Complexity

The use of multiple security tools in one CI/CD pipeline creates challenges because of the architecture [10]. These tools include SAST, SCA, DAST, and malware detection through GNN and have their roles in the security process and must therefore be coordinated [4]. The tools used in the security process are offered the necessary input, output, and algorithms [11]. It should also be pointed out that the combination of different analysis methods in one CI/CD pipeline means that there must be a compilation of all findings into one report [9].

ii). Computational Overhead

The variety of security techniques adds extra complexity to the CI/CD pipeline owing to high computing costs. Different kinds of algorithms, for example, code static analysis, software dynamic testing, dependency scanning, and so forth, are used in each step [5]. This makes the CI/CD workflow costly and ineffective, especially since triggering the pipeline may delay software deployment [11]. However, effective hardware usage becomes necessary whenever the load is huge within the system during security activities within the CI/CD pipeline.

iii). Resource Requirements for GNN Models

The application of GNNs for detecting malware consumes significant computational resources than others [1]. This is due to the use of graphs by GNNs to describe the various associations in binaries [8]. The training of GNNs requires extensive data and sophisticated devices such as GPUs [12, 13]. The implementation of GNNs' malware detection process into an actual-time CI/CD pipeline may face difficulty, particularly with restricted computational resources [15]. High memory usage and computation time may be among them. Thus, some optimization strategies are crucial [12].

iv). Real-Time Detection and Accuracy

One of the other major issues in relation to CI/CD pipeline security is proper identification of any potential attacks that can happen. As such, rapid detection is critical because delay is not desirable. However, one must ensure that this process does not lead to compromising the quality of the analysis conducted [3].

It is equally important to come up with a means of dealing with both false positives and false negatives, with the former blocking genuine code while the latter allowing for delivery of malicious software [5].

4.2 Future Scope

Future research might focus on developing a model that is optimized and lightweight, and at the same time, has lower computing power requirements but remains accurate in detection [12][13]. Optimization and graph-based representation of the model will help ensure the appropriateness of the malware detection system in the discussed CI/CD environment [1][15]. Moreover, giving the model adaptive learning capabilities will make sure that it constantly updates itself and evolves based on changing threats online [5]. Another possible future research could also focus on integrating a full-scale decision-making algorithm into the CI/CD process [6]. With the use of appropriate AI tools, the system will be able to analyze the security report and take necessary action based on it – this includes either approving or declining code build requests, as well as suggesting any mitigating strategies [2]. This would add automation to the process, increasing efficiency [3]. Additional features could be introduced to help tackle the issue of scalability [4].

5. CONCLUSION

One of the key conclusions drawn by this study is the difference that has been revealed between the rapidity with which the software gets delivered with the help of current technological solutions and the inability of many security tools to detect and disable new threats. In this case, the necessity of applying intelligent security approaches that could be effective at every stage of development and distribution comes to light. Automation has become an industry standard when it comes to software distribution; however, security solutions have not yet reached adequate levels of development. In order to bridge this gap, the Zentronyx platform is recommended. This platform allows combining the traditional means of securing software with analytics, offering the means to analyze and interpret software artifacts based on the technology of Graph Neural Networks. Integration of popular techniques like Static Application Security Testing (SAST), Software Composition Analysis (SCA), and Dynamic Application Security Testing (DAST) with this analytics tool offers more options for analysis and identifying problems with software artifacts. For example, one would be able to analyze complex binary structures, which would have been harder otherwise with the traditional means of securing software. Centralized management, policy-driven decisions, and immediate risks assessment contribute to the efficiency of the whole process. All these aspects improve the precision of detecting vulnerabilities and allowing developers and security experts to make better decisions faster. To sum up, the presented work demonstrates the importance of adopting a sophisticated and unified strategy in protecting software. Future improvements can focus on optimizing the platform in terms of performance, compatibility.

REFERENCE

[1] Alshehri, S. M., Sharaf, S. A., & Molla, R. A. (2025). Systematic Review of Graph Neural Network for Malicious Attack Detection. Information, 16(6), 470. https://doi.org/10.3390/info16060470

[2] Aminat Bolaji Bello, Akeem Olakunle Ogundipe, Awobelem A. George, & Olabode Anifowose. (2025). The role of AI and machine learning in cybersecurity: Advancements in threat detection, anomaly detection and automated response. *International Journal of Science and Research Archive*, *14*(2), 1587–1597. https://doi.org/10.30574/ijsra.2025.14.2.0542

[3] Binbeshr, F., & Imam, M. (2025). *Comparative Analysis of AI-Driven Security Approaches in DevSecOps: Challenges, Solutions, and Future Directions*. arXiv. https://doi.org/10.48550/ARXIV.2504.19154

[4] Carlos Bautista Ramos, R., & Yoo, S. G. (2025). Cybersecurity in DevOps Environments: A Systematic Literature Review. *IEEE Access*, *13*, 191959–191979. https://doi.org/10.1109/ACCESS.2025.3582892

[5] Joshi, C., Kumar, J., & Kumawat, G. (2025). Detection of unseen malware threats using generative adversarial networks and deep learning models. *Scientific Reports*, *15*(1), 34804. https://doi.org/10.1038/s41598-025-18811-3

[6] Justin Rajakumar Maria Thason. (2025a). AI-Driven DevSecOps: Advancing Security and Compliance in Continuous Delivery Pipelines. *International Research Journal on Advanced Engineering and Management (IRJAEM)*, *3*(05), 1666–1673. https://doi.org/10.47392/IRJAEM.2025.0268

[7] Kaniewski, S., Schmidt, F., Enzweiler, M., Menth, M., & Heer, T. (2025). *A Systematic Literature Review on Detecting Software Vulnerabilities with*

Large Language Models. arXiv. https://doi.org/10.48550/ARXIV.2507.22659

[8] Kulkarni, P., & OShaughnessy, S. (2025). Malware Detection Using Dynamic Graph Neural Networks. *European Conference on Cyber Warfare and Security*, *24*(1), 830–837. https://doi.org/10.34190/eccws.24.1.3459

[9] Alshehri, S. M., Sharaf, S. A., & Molla, R. A. (2025). Systematic Review of Graph Neural Network for Malicious Attack Detection. *Information*, *16*(6), 470. https://doi.org/10.3390/info16060470

[10] Aminat Bolaji Bello, Akeem Olakunle Ogundipe, Awobelem A. George, & Olabode Anifowose. (2025). The role of AI and machine learning in cybersecurity: Advancements in threat detection, anomaly detection and automated response. *International Journal of Science and Research Archive*, *14*(2), 1587–1597. https://doi.org/10.30574/ijsra.2025.14.2.0542

[11] Kulkarni, P., & OShaughnessy, S. (2025). Malware Detection Using Dynamic Graph Neural Networks. *European Conference on Cyber Warfare and Security*, *24*(1), 830–837. https://doi.org/10.34190/eccws.24.1.3459

[12] Mohammed, K., Shanmugam, B., & El-Den, J. (2025). Evolution of DevSecOps and Its Influence on Application Security: A Systematic Literature Review. *Technologies*, *13*(12), 548. https://doi.org/10.3390/technologies13120548

[13] Shokouhinejad, H., Razavi-Far, R., Mohammadian, H., Rabbani, M., Ansong, S., Higgins, G., & Ghorbani, A. A. (2025). *Recent Advances in Malware Detection: Graph Learning and Explainability*. arXiv. https://doi.org/10.48550/ARXIV.2502.10556

[14] Sood, A. K., & Zeadally, S. (2025). Malicious AI Models Undermine Software Supply-Chain Security. *Communications of the ACM*, 3704724. https://doi.org/10.1145/3704724

[15] Tarapata, Z., & Romańczuk, J. (2025). Some Improvements of Behavioral Malware Detection Method Using Graph Neural Networks. *Applied Sciences*, *15*(21), 11686. https://doi.org/10.3390/app152111686