Research Article

International Journal of Multidisciplinary Applied and Science Research
Volume 01, Issue 06, PP. 252–258
e.ISSN: 3090 2789

Highlighting Significance and Contribution: Five Recent Studies on Static Analysis in Software Engineering and Cybersecurity

Muhammad Hafiz Firmansyah[1], Ilham Zainuri[2], & Fachrudin Pakaja[3]*

[1,2,3] Faculty of Engineering and Informatics, Gajayana University, Malang, Indonesia,
Email: hafizsengkaling@gmail.com [1], ilhamzainuri010404@gmail.com [2]. fachrudinpakaja@unigamalang.a c.id [3].

*Corresponding Author Email: hafizsengkaling@gmail.com

Received: October 16, 2025
Revised: October 06, 2025
Accepted: November 15, 2025

Abstract
Static testing has become a critical approach in ensuring the quality, security, and reliability of software systems. Recent developments include the application of machine learning, abstract interpretation, and query-based approaches to improve the effectiveness of analysis. Objectives and methods: This systematic review aims to consolidate and analyse findings from five recent studies (2024-2025) on static testing methods across various domains, including source code analysis, vulnerability detection, Android malware detection, business process modelling, and data leakage prevention in machine learning. The methods used were thematic and comparative analysis of the contributions, methodologies, and limitations of each study. Main results: The synthesis shows that static testing approaches are increasingly integrating dynamic techniques, machine learning, and formal analysis to address the complexity of modern systems. However, challenges such as limited coverage, the need for industry validation, and computational complexity remain obstacles. Conclusions and implications: Static testing continues to evolve with hybrid and data-driven approaches. Further research is needed that focuses on expanding coverage, integrating industrial pipelines, and improving accessibility for practitioners.

Keyword: Static Analysis, Software Testing, Software Engineering, Cybersecurity.

1. Introduction

In a digital era driven by increasing system complexity and evolving security threats, static testing is an essential foundation for detecting vulnerabilities (Goseva-Popstojanova & Perhinschi, 2015; Pistoia et al., 2007), errors (Alghamdi & Eassa, 2019; Ayewah et al., 2008; Zheng et al., 2006), and nonconformities from the development stage (Kikuchi & Kikuno, 2001). In the contemporary software development paradigm, static testing has transformed its role from merely a bug detection tool to a strategic component in the Software Development Life Cycle (SDLC) (Kleidermacher, 2008; Saeed et al., 2025; Teja Swaroop, 2025). The use of static analysis methods allows software developers to identify security vulnerabilities, logical inconsistencies, and deviations from code standards early on without requiring program execution (Goseva-Popstojanova & Perhinschi, 2015; Pistoia et al., 2007; Shahriar & Zulkernine, 2012).

The evolution of static testing system development is currently driven by three main factors: first, the increasing complexity of software architecture (microservices, distributed systems) (Thomson, 2021; Knodel et al., 2006); second, the proliferation of increasingly sophisticated cyber threats (Shinde & Ardhapurkar, 2016; Mazzolin & Samueli, 2020), and third, increasingly stringent regulatory and industry standards (Pargaonkar, 2023; Wild & Knapp, 2017). Traditional approaches are often inadequate for handling the dynamics of modern systems, giving rise to innovations such as graph machine learning (GML) (Kanewala et al., 2016), query-based static analysis (Q-SA) (Gebhart, 2014), abstract interpretation (Cousot & Cousot, 2000), and the integration of static-dynamic techniques (Dano, 2022; Salihu et al., 2019). However, these developments are scattered across various domains without a cohesive synthesis.

This paper aims to: (1) review and synthesise findings from five recent studies on static testing methods, (2) identify trends, strengths, and limitations of existing approaches, and (3) provide recommendations for future research and practice. Although conventional static testing approaches currently face fundamental challenges in dealing with the dynamic nature of modern applications, increasingly advanced obfuscation techniques, and the need for analysis that can adapt to iterative code changes. However, the main contributions of this review paper are to (1) map the current landscape of static testing research, (2) provide a comparative analysis of methodologies and study results, and (3) identify research gaps and opportunities for cross-domain collaboration.

This paper is organized as follows: section 1 explains the background, significance, and objectives of this study; section 2 explains the review methodology; section 3 presents the results of the thematic synthesis; section 4 discusses the implications and limitations; and section 5 concludes and provides directions for future research.

2. The Art of Research

This study adopts a systematic narrative review as the primary research design, chosen to ensure thematic relevance and meaningful methodological contributions in the domains of software engineering and security (Best et al., 2014; Dehkordi et al., 2021). The research framework is structured to systematically identify, evaluate, and synthesize recent scholarly work on static testing models in modern applications and software obfuscation techniques.

3. Method

This review uses a systematic narrative review approach selected based on thematic relevance and methodological contribution (Best et al., 2014; Dehkordi et al., 2021). The main themes addressed in this study are static testing models in modern applications and obfuscation techniques within the scope of models in the domain of software engineering and security. This study uses five journal data sources taken from SCOPUS with a time range of 2024-2025 (see Table 1). The analysis procedure in this study is thematic, grouping the findings into the following categories: (1) motivation and context, (2) methodology, (3) contributions, (4) main results, (5) limitations, and (6) recommendations (Fong, 2009). We used evaluation criteria (Kitchenham et al., 2009) based on the significance of the problem, validity of claims, scientific contribution, and practical relevance. This study used comparative analysis and qualitative synthesis tools to identify patterns, gaps, and trends across studies.

Table 1. Journal Name Summary

No	Author	Title	Journal Name	Tier Journal
1	Christfort, A. K., Cosma, V. P., Debois, S., Hildebrandt, T. T., & Slaats, T. (2025)	Static and dynamic techniques for iterative test-driven modelling of Dynamic Condition Response Graphs	Data and Knowledge Engineering	Q-2
2	Urban, C., Subotić, P., & Drobnjaković, F. (2025)	Static Analysis by Abstract Interpretation Against Data Leakage in Machine Learning	Science of Computer Programming	Q3
3	Li, Z., Liu, Z., Wong, W. K., Ma, P., & Wang, S. (2024)	Evaluating C/C++ Vulnerability Detectability of Query-Based Static Application Security Testing Tools	IEEE Transactions on Dependable and Secure Computing	Q-1
4	Maarleveld, J., Guo, J., & Feitosa, D. (2025)	A systematic mapping study on graph machine learning for static source code analysis	Information and Software Technology	Q1
5	Molina-Coronado, B., Ruggia, A., Mori, U., Merlo, A., Mendiburu, A., & Miguel-Alonso, J. (2025)	Light up that Droid! On the effectiveness of static analysis features against app obfuscation for Android malware detection	Journal of Network and Computer Applications	Q1

4. Result

a. Key Findings

Based on the analysis of the five journals, four main themes can be identified in the development of static testing methods:

1. The application of Machine Learning and Graph-Based Analysis studies by Maarleveld et al. (2025) shows the dominance of Graph Neural Networks (GNN) in static source code analysis, while Molina-Coronado et al. (2025) developed an ML-based malware detector that is resistant to obfuscation. This trend indicates a shift from rule-based approaches to data-driven learning.
2. Integration of Static and Dynamic Techniques by Christfort et al. (2025) proposes the integration of static verification and dynamic alignment checking for open test-driven modelling validation, demonstrating the effectiveness of hybrid approaches in reducing validation complexity.
3. The use of Abstract Interpretation and Formal Methods by Urban et al. (2025) applies abstract interpretation to detect data leakage in machine learning, providing a formal foundation for proactive analysis that previously relied on retroactive detection.
4. Evaluation and Benchmarking Tools by Li et al. (2024) develops C/C++ vulnerability benchmarks and SAST-MT tools for evaluating Q-SAST tools, highlighting the importance of evaluation standardization in static testing research.

b. Analysis of Research Aspects

The synthesis of the five studies is divided based on several items (see Table 2). Based on the findings summarised in the table, it can be concluded that these five studies collectively describe current trends and challenges in the field of static analysis applied to various domains, ranging from business process modelling, machine learning security, software vulnerability detection, to graph-based code analysis and Android security. The motivation behind each study stems from pressing practical needs in the industry, such as the need for more adaptive process modelling, prevention of data leaks in ML pipelines, evaluation of automated security tools, mapping of the rapidly evolving technology landscape, and handling of increasingly sophisticated code obfuscation techniques. In terms of methodology, these studies demonstrate a diversity of approaches,

ranging from systematic mapping studies and formal analysis based on abstract interpretation to large-scale empirical experiments and real-world case studies, reflecting an effort to bridge the gap between theory and practice with a strong methodological foundation.

The main contribution of this series of studies lies in providing an innovative theoretical and practical foundation for solving specific problems in static analysis. The first and second studies, for example, offer integrative solutions by combining static and dynamic techniques for declarative model validation and applying formal analysis to proactively detect data leaks, as demonstrated by a 60% increase in detection. Meanwhile, the third and fifth studies contribute empirical evaluations and tools (benchmarks, datasets, and detectors) that can be directly utilized by the research and industry communities to improve the reliability of vulnerability and malware detection. The fourth study serves as a comprehensive roadmap that identifies trends, dominant techniques, and research gaps in the application of graph machine learning, thereby guiding future research to more productive areas and avoiding duplication.

However, these findings also reveal a number of limitations that are consistent across all studies, such as the scope of evaluation being limited to specific datasets or tools, the lack of empirical validation in actual industrial pipelines, and methodological complexities that can hinder adoption by practitioners. These limitations indicate that although the scientific contributions and practical relevance of each study have been significant, further efforts are needed to transform academic findings into truly mature solutions that are ready for integration into production environments. Therefore, recommendations for future research should focus on expanding the scope of evaluation, closer collaboration with industry, simplifying implementation, and developing more robust quantitative metrics to measure the effectiveness and real impact of the proposed static analysis techniques.

Table 2. Summary of Findings Based on Research Aspects

Paper ID	Motivation & Context	Methodology	Contribution	Main Results	Limitations
1	Combining the flexibility of declarative models with the readability of imperative models in TDM.	Three stages: benchmarking, static verification, alignment checking. Real-world case studies.	Expanding the theory of open test-driven modelling with the integration of static-dynamic techniques.	The combination of techniques improves the validation of open tests, supporting iterative development.	Limited to DCR Graphs, not yet tested in industrial pipelines.
2	Preventing data leakage from the ML development stage, especially in interactive notebooks.	Static analysis based on abstract interpretation. Dataset of 2111 Kaggle notebooks.	A formal framework for detecting data leaks before model training.	93% precision, detection of 25 real cases, a 60% improvement over previous methods.	Focus on train- test contamination, dataset only Kaggle.
3	Evaluating the effectiveness of Q-SAST in detecting C/C++ vulnerabilities.	Benchmarking CVE/CWE, analyzing bug-fixing PRs, SAST-MT tools.	First empirical study on Q-SAST, vulnerability benchmarking, open-source SAST-MT tools.	Q-SAST is flexible but not yet optimal, with many false positives/negatives.	Limited to CodeQL, descriptive analysis.

4	No systematic mapping of GML implementation for static code analysis.	Systematic Mapping Study (SMS) with 323 primary studies.	Comprehensive mapping of GML, identification of 7 sub-domains, classification of artifacts/models.	GML has been widely used since 2018, but few dominant techniques.	Descriptive, no empirical validation, limited search in Scopus.
5	Obfuscation disrupts malware detection based on static analysis.	Experiments with a dataset of 95,000 APKs, evaluation of features vs. obfuscation.	Evaluation of feature sensitivity, obfuscation-resistant detectors, open dataset.	Reflection & encryption most damaging to features, new detectors more resistant.	Conceptual, not yet fully validated, limited literature coverage.

c. Evaluation of Research Criteria

Based on the evaluation of the five studies (see Table 3), the findings show that the significance of the issues raised is high to very high, reflecting the urgency and relevance of these topics in the current academic and industrial context. Issues such as the need for adaptive process modelling, the danger of data leaks in machine learning, the evaluation of the reliability of automated security tools, the mapping of the rapidly developing field of graph machine learning, and the threat of code obfuscation in Android applications are considered critical issues that require innovative solutions. The high practical relevance of each study reinforces its status as a contributor that is not only theoretical but also practically applicable. For example, research on data leak detection and Android malware detection directly offers frameworks and tools that can be integrated into existing development pipelines, while systematic mapping for graph machine learning serves as a strategic guide for researchers and industry developers.

In terms of claim validity, the majority of studies are supported by robust and transparent methodologies, such as real-world case studies, formal analysis based on abstract interpretation, experiments with large-scale datasets, and rigorous systematic mapping procedures. This provides a credible basis for the claims made. However, there is variation in the level of validity, with some studies being rated as "moderately valid" due to limitations in the scope of the tools or datasets, even though the overall methodology remains clear and structured. The scientific contributions are rated as significant to very significant, with each study presenting new approaches, knowledge synthesis, or tools that enrich the scientific foundation in their respective fields. For example, the combination of static-dynamic techniques for declarative modelling, a proactive approach to detecting data leaks, and the first empirical study of query-based SAST tools are considered important steps forward for the research community.

Overall, these findings underscore that the five studies not only successfully identified and addressed significant issues, but also did so with a high degree of validity and scientific contribution, as well as clear practical relevance. This combination makes them examples of how research in the field of static analysis and software security can bridge the gap between theory and practice. The collective strength of these studies lies in their ability to provide solutions that are methodologically grounded, conceptually innovative, and ready for industry adoption, thereby strengthening the research and development ecosystem in the face of ever-evolving technological challenges.

Table 3. Evaluation Based on Research Criteria

Paper ID	Significance of the Problem	Claim Validity	Scientific Contribution	Practical Relevance	Significance of the Problem
1	High: Addresses the need for adaptive TDM in business processes.	Valid: Supported by case studies and structured methods.	Significant: Combines static-dynamic approaches in declarative modelling.	High: Applicable in cross-organizational case management systems.	High: Addresses the need for adaptive TDM in business processes.
2	Very High: Data leakage is a critical	Highly Valid: Supported by formal theory	Very Significant: Presents a proactive approach with	Very High: Can be directly integrated into ML pipelines	Very High: Data leakage is a

	issue in production ML.	and extensive empirical evaluation.	abstract interpretation.	and data science notebooks.	critical issue in production ML.
3	High: It is important to assess the reliability of code security tools ( ).	Sufficiently Valid: Limited to CodeQL , but the methodology is clear.	Significant: The first empirical study of with benchmarks and open tools.	High: Helps practitioners choose and improve SAST tools.	High: Important for evaluating the reliability of code security tools.
4	High: Necessary for mapping a rapidly evolving field.	Valid: SMS with rigorous methodology and extensive data.	Highly Significant: Provides a comprehensive research map and recommendations for research directions.	High: Serves as a guide for researchers and industry in adopting GML.	High: Essential for mapping rapidly evolving fields.
5	Very High: Obfuscation is a real threat to Android security.	Valid: Supported by large datasets and in- depth experiments.	Significant: Provides in-depth analysis of the impact of obfuscation and more robust detectors.	Very High: Directly useful for antivirus developers and security researchers.	Very High: Obfuscation is a real threat to Android security.

5. Discussion

a. Paradigmatic Evolution in Static Testing

The analysis of these five journals collectively indicates a paradigmatic evolution in the field of static testing, moving from isolated and reactive approaches toward integrated, proactive, and data-driven frameworks. The dominance of Machine Learning applications (particularly GNN) and Abstract Interpretation marks two sides of this transformation: on one hand, data-driven approaches such as ML offer adaptability to the complexity and variability of modern code (including obfuscated malware), while on the other hand, formal methods like abstract interpretation provide a solid mathematical foundation for analyses whose correctness can be guaranteed, such as in data leakage detection. This shift not only enhances detection capabilities but also transforms the nature of analysis from mere pattern matching (rule-based) to a deeper contextual understanding of structure and data flow within software.

Amidst these methodological advances, two other themes emerge as critical counterbalances: integration and evaluation. The proposed integration of static and dynamic techniques, as demonstrated by Christfort et al. (2025), recognizes that hybrid approaches are often necessary to capture the complexity of system behavior comprehensively, where static validation can be enriched and verified with dynamic checks. However, the proliferation of these new methods and tools would lose its meaning without reliable evaluation standards. This is where work such as that done by Li et al. (2024) in providing benchmarks and standardized evaluation frameworks becomes important. These efforts form the backbone for objective progress in the field, enabling fair comparisons, identifying tool weaknesses, and ultimately driving improvements in the quality and reliability of static application security testing (SAST) tools themselves, so that methodological innovation can be directed toward solving measurable practical problems.

Material discussion mainly concerned, whether the results are consistent with the hypothesis or not, and put forward the argument.

b. Trends, Strengths, and Limitations of the Approach

The synthesis of these five studies confirms that the field of static analysis is undergoing a productive period of transformation, driven by concrete industrial needs and enabled by methodological advances. The diversity of approaches—from systematic mapping to large-scale empirical experiments—is not merely variation, but a necessary response to the complexity of different target domains, ranging from business process modelling to machine learning security. The collective significance of these findings lies in the concerted effort to bridge the traditional gap between theory and practice. The contributions, whether in the form of formal foundations (such as abstract interpretation), integrative solutions (static-dynamic hybrids), or evaluation infrastructure (benchmarks and datasets), form a layered and complementary foundation. This foundation not only advances technical detection and analysis capabilities, such as significant improvements in data leak detection accuracy, but also provides a valuable roadmap, as produced by mapping studies, to direct future research investment to the most promising areas and avoid duplication of effort.

However, discussion of these contributions must be balanced with critical reflection on the limitations that have been

consistently revealed. The gap between sophisticated methodological innovations and their readiness for adoption in actual industrial pipelines remains a major challenge. Limitations such as the scope of evaluation on specific datasets, lack of validation in real production environments, and implementation complexity suggest that the maturity of a technique is not only measured by its academic results, but also by its ease of integration and resilience to real-world variability. Therefore, this discussion leads to an imperative for further research: that the ultimate value of developing static analysis methods lies in their practical impact. Future recommendations need to explicitly shift the focus from simply proving the concept to engineering robust solutions. This requires symbiotic collaboration with industry from the design phase, simplification of interfaces and implementation, and the development of new evaluation metrics that measure not only accuracy, but also efficiency, scalability, and integration costs in the actual software development cycle.

a. Recommendations for Future Research and Practice

Based on evaluations that demonstrate high scientific validity and significance, these studies open up several strategic avenues for future research. First, efforts are needed to consolidate and generalize specific findings that are still limited to certain datasets or tool contexts. Future research should be designed to validate frameworks such as data leakage detection or static-dynamic integration in more diverse and complex environments, approximating actual industrial pipeline conditions. Second, transdisciplinary collaboration between static analysis experts, machine learning practitioners, and industrial software engineers needs to be intensified. This is important to address implementation challenges and develop new evaluation metrics that measure not only technical accuracy, but also usability, integration costs, and the economic impact of adopting a tool. In addition, the roadmap from the systematic mapping study should be used as a reference to explore unexplored research gaps, such as the application of static analysis in the domain of edge computing or autonomous systems.

On the practical side, findings with high relevance and application readiness, such as vulnerability benchmarks, Android malware detectors, and data leak detection frameworks, require structured adoption mechanisms. Industry organizations are advised to initiate pilot projects integrating these tools or methodologies into their DevSecOps pipelines, while contributing real-world data and feedback for further refinement. Standard framework developers and certification bodies can leverage empirical findings from tool evaluation studies (benchmarking) to develop guidelines and objective criteria for selecting Static Application Security Testing (SAST) tools, thereby reducing reliance on vendor claims. Finally, academic and professional training institutions need to incorporate core concepts from this research, such as abstract interpretation, graph-based analysis, and hybrid approaches, into their curricula to build talent capable of translating academic innovation into robust industry solutions, thereby enabling a faster and more effective cycle between research and practice.

6. Conclusion

A review of five recent studies on static testing methods reveals that modern approaches are increasingly adopting machine learning techniques, formal analysis, and static-dynamic integration to improve effectiveness and accuracy. Key contributions include research mapping, evaluation tools, datasets, and frameworks that support replication and further development. Static testing is no longer just static code analysis, but has evolved into a complex and multidisciplinary methodological ecosystem. Cross-domain collaboration between software security, machine learning, and process modelling can accelerate innovation.

Acknowledgments

-

References

1. Alghamdi, A. M., & Eassa, F. E. (2019). OpenACC errors classification and static detection techniques. IEEE Access, 7, 113235-113253.
2. Ayewah, N., Pugh, W., Hovemeyer, D., Morgenthaler, J. D., & Penix, J. (2008). Using static analysis to find bugs. IEEE software, 25(5), 22-29.
3. Best, P., Manktelow, R., & Taylor, B. (2014). Online communication, social media and adolescent wellbeing: A systematic narrative review. Children and Youth Services Review, 41, 27-36.
4. Christfort, A. K. F., Cosma, V. P., Debois, S., Hildebrandt, T. T., & Slaats, T. (2025). Static and dynamic techniques for iterative test-driven modelling of Dynamic Condition Response Graphs. Journal of Systems and Software, 183, 107722.
5. Cousot, P., & Cousot, R. (2000, August). Abstract interpretation based program testing. In Proceedings of the SSGRR 2000 Computer & eBusiness International Conference (pp. 161-192). L'Aquila, Italy: Scuola Superiore G. Reiss Romoli.
6. Dano, E. B. (2022, July). Systems Engineering Integration and Test Challenges due to Security Measures in a Cloud-Based System. In INCOSE International Symposium (Vol. 32, No. 1, pp. 224-232).
7. Dehkordi, A. H., Mazaheri, E., Ibrahim, H. A., Dalvand, S., & Gheshlagh, R. G. (2021). How to write a systematic review: A narrative review. International journal of preventive medicine, 12(1), 27.
8. Fong, P. W. (2009). Reading a computer science research paper. ACM SIGCSE Bulletin, 41(2), 138-140.
9. Gebhart, M. (2014). Query-based static analysis of web services in service-oriented architectures. International Journal on Advances in Internet Technology Volume 7, Number 1 & 2, 2014.

10. Goseva-Popstojanova, K., & Perhinschi, A. (2015). On the capability of static code analysis to detect security vulnerabilities. Information and Software Technology, 68, 18-33.
11. Kanewala, U., Bieman, J. M., & Ben-Hur, A. (2016). Predicting metamorphic relations for testing scientific software: a machine learning approach using graph kernels. Software testing, verification and reliability, 26(3), 245-269.
12. Kikuchi, N., & Kikuno, T. (2001, December). Improving the testing process by program static analysis. In Proceedings Eighth Asia-Pacific Software Engineering Conference (pp. 195-201). IEEE.
13. Kitchenham, B., Brereton, O. P., Budgen, D., Turner, M., Bailey, J., & Linkman, S. (2009). Systematic literature reviews in software engineering–a systematic literature review. Information and software technology, 51(1), 7-15.
14. Kleidermacher, D. N. (2008, May). Integrating static analysis into a secure software development process. In 2008 IEEE Conference on Technologies for Homeland Security (pp. 367-371). IEEE.
15. Knodel, J., Lindvall, M., Muthig, D., & Naab, M. (2006, March). Static evaluation of software architectures. In Conference on Software Maintenance and Reengineering (CSMR'06) (pp. 10-pp). IEEE.
16. Li, Z., Liu, Z., Wong, W. K., Ma, P., & Wang, S. (2024). Evaluating C/C++ Vulnerability Detectability of Query-Based Static Application Security Testing Tools. IEEE Transactions on Software Engineering, 50(3), 455-470.
17. Maarleveld, J., Guo, J., & Feitosa, D. (2025). A systematic mapping study on graph machine learning for static source code analysis. Information and Software Technology, 183, 107722.
18. Mazzolin, R., & Samueli, A. M. (2020, August). A survey of contemporary cyber security vulnerabilities and potential approaches to automated defence. In 2020 IEEE International Systems Conference (SysCon) (pp. 1-7). IEEE.
19. Molina-Coronado, B., Ruggia, A., Mori, U., Merlo, A., Mendiburu, A., & Miguel-Alonso, J. (2025). Light up that Droid! On the effectiveness of static analysis features against app obfuscation for Android malware detection. Journal of Network and Computer Applications, 235, 104094.
20. Pargaonkar, S. (2023). Advancements in security testing: A comprehensive review of methodologies and emerging trends in software quality engineering. International Journal of Science and Research (IJSR), 12(9), 61-66.
21. Pistoia, M., Chandra, S., Fink, S. J., & Yahav, E. (2007). A survey of static analysis methods for identifying security vulnerabilities in software systems. IBM systems journal, 46(2), 265-288.
22. Saeed, H., Shafi, I., Ahmad, J., Khan, A. A., Khurshaid, T., & Ashraf, I. (2025). Review of Techniques for Integrating Security in Software Development Lifecycle. Computers, Materials & Continua, 82(1).
23. Salihu, I. A., Ibrahim, R., Ahmed, B. S., Zamli, K. Z., & Usman, A. (2019). AMOGA: A static-dynamic model generation strategy for mobile apps testing. IEEE Access, 7, 17158-17173.
24. Shahriar, H., & Zulkernine, M. (2012). Mitigating program security vulnerabilities: Approaches and challenges. ACM Computing Surveys (CSUR), 44(3), 1-46.
25. Shinde, P. S., & Ardhapurkar, S. B. (2016, February). Cyber security analysis using vulnerability assessment and penetration testing. In 2016 World Conference on Futuristic Trends in Research and Innovation for Social Welfare (Startup Conclave) (pp. 1-5). IEEE.
26. Teja Swaroop, A. (2025). The Transformative Impact of Artificial Intelligence on Professional Software Development: A Comprehensive Analysis. Title of Paper: The Transformative Impact Of Artificial Intelligence On Professional Software Development: A Comprehensive Analysis published in Page No, 13(8), b74-b90.
27. Thomson, P. (2021). Static Analysis: An Introduction: The fundamental challenge of software engineering is one of complexity. Queue, 19(4), 29-41.
28. Urban, C., Subotić, P., & Drobnjaković, F. (2025). Static Analysis by Abstract Interpretation Against Data Leakage in Machine Learning. Proceedings of the ACM on Programming Languages, 9(PLDI), 1-25.
29. Wild, C. L., & Knapp, J. E. (2017). Standards in the Testing Industry. In Improving Testing (pp. 59-81). Routledge.
30. Zheng, J., Williams, L., Nagappan, N., Snipes, W., Hudepohl, J. P., & Vouk, M. A. (2006). On the value of static analysis for fault detection in software. IEEE transactions on software engineering, 32(4), 240-253.