International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Secure Software Development: Problems and Solutions

Manal Jaza Al Anzi[1], Maha Abdul-Rahman Al Balwi[2], Dr. OnytraAbbass*[3]

Submitted:11/03/2024 **Revised**: 26/04/2024 **Accepted**: 03/05/2024

Abstract: Secure software development has become progressively dire in the face of increasing cyber threats and the mounting dependence on digital systems across diverse sectors. This paper presents an inclusive review of the problems and answers in the field of developing secure software, drawing insights from a systematic literature review of peer-reviewed articles available within the period from 2015 to 2024. The study recognises key hindrances in adopting secure development practices, comprising having security integrated into the lifecycle of developing software, tackling evolving technologies' security implications, and bridging the skills gap in the industry. The paper investigates effective methodologies such as Security Development Lifecycle (SDL), DevSecOps, and advanced testing techniques like Dynamic Application Security and Testing (DAST). Static Application Security Testing (SAST) and Findings emphasize the importance of a comprehensive approach to secure software development, comprising organizational culture, constant education, and the implementation of security-focused frameworks. The research also highlights promising trends in automation, AI-assisted security analysis, and cloud-native security approaches. The present paper adds to the literature of developing secure software via tackling current challenges, assessing current solutions, and suggesting future guidelines for research and practice. The insights provided are worthy for designers of software, security specialists, and corporations struggling to heighten the secure development abilities in a progressively complex digital landscape.

Keywords:*secure software development, cybersecurity, threat modelling, software security testing, security development lifecycle.*

1. Introduction

Generally speaking, developing software is the procedure adopted to construct software. Nowadays, software development is a more complex process than ever was and it encounters the challenges, where security has become one of the most critical.

The security issues and recognizing the weaknesses, hazards have become an inseparable component of software engineering. Secure software development is a method (often associated with DevSecOps) for establishing software that incorporate security within each point of (SDLC) software development life cycle [1]. Secure software development has become a dire concern in the quickly progressing landscape of information technology.

Since cyber risks continually increase in sophistication as well as frequency, the need for forceful safety procedures incorporated into the software development lifecycle has never been more predominant. It is within highly incorporated technology settings that information security is becoming a pivotal point for designing, developing and deploying software applications. Guaranteeing a great deal of confidence in the security and quality of these applications is necessary to their final success. Information security has therefore become a fundamental requirement for software applications, driven by the need to guard critical assets and the need to create and maintain widespread trust in computing [2].

The increasing dependence on digital systems across different areas, including healthcare, finance and government, has raised the possible effect of security violations. According to a report by Accenture [3], the mean expense of cybercrime for a corporation reached $13 million in 2018, a 12% increase from the previous year. This statistic underlines the pressing need for improved secure software development practices. Furthermore, the emergence of evolving technological devices as AI, cloud computing, and (IoT), has introduced new weaknesses and attack courses. A study by Gartner [4] expects that by 2025, 75% of business-critical applications will be running on cloud platforms, underlining the need for cloud-native approaches to guarantee security in software development.

Driven by what is mentioned above and based on the significance of Secure software development to encounter increasing cyber threats, this paper aims to explore the primary challenges faced in secure software development and propose effective solutions to address these issues.

Driven by what is mentioned above and based on the significance of Secure software development to encounter increasing cyber threats, this paper aims to explore the primary challenges faced in secure software development and propose effective solutions to address these issues.

1.1. **This paper will address the coming key questions:**

[1] *Department of Information Technology. University Of Tabuk –71411, KSA*
ORCID ID :0009-0004-4100-0224
Email:Manal @gmail.com
[2] *Department of Information Technology. University Of Tabuk –71411, KSA*
ORCID ID : 0009-0004-5201-0206
Email: Maha@gmail.com
[3] *Department of Computer Science. University Of Tabuk –71411, KSA*
** Corresponding Author Email: obashir@ut.edu.sa*

1.2. 1. What are the primary challenges in implementing secure software development practices?

1.3. 2. How can organizations effectively integrate security into their software development lifecycle?

1.4. 3. What emerging technologies and methodologies show promise in enhancing software security?

1.5. 4. How can the industry bridge the skills gap in secure software development?

By examining these questions through a comprehensive literature review, the current paper aims to contribute to the existing knowledge in secure software development and give practical perceptions for specialists and academics alike.

2. Method

This study uses a methodical review of literature as the principal method of collecting and analysing data. The systematic literature review methodology was chosen for its rigorous and transparent approach to synthesizing existing research, as outlined by Kitchenham and Charters[5]. The literature search was conducted using some academic databases such as ACM Digital Library, ScienceDirect, SpringerLink and Google Scholar. Keywords used in the search included combinations of the following terms: "Secure software development", "Software security", "Secure coding practices", "Security Development Lifecycle (SDL)", "DevSecOps" and "Threat modelling".The systematic literature review revealed several key themes regarding the problems and solutions in secure software development. These findings are categorized into major challenges and corresponding solutions or best practices.required more refined preprocessing.

3. Major Challenges in Secure Software Development and Their Solutions

3.1. Integration of Security into the Software Development Lifecycle (SDLC)

Incorporating security into the Software Development Life Cycle (SDLC) is fundamental to guard applications from vulnerabilities and threats. Many organizations strive to effortlessly integrate methods heightening security through the SDLC. A study by [6] found that 68% of surveyed companies reported difficulties in applying security measures at every stage of development. This often results in security being treated as an afterthought, leading to weaknesses that are expensive and time-consuming to address later in the development process. To encounter such challenge, implementation of SDL frameworks has shown significant improvements in software security. Microsoft's case study [7]reported a 50% reduction in security vulnerabilities after widespread adoption of their SDL practices, which is very encouraging. This success aligns with earlier findings by McGraw [8], who emphasized the necessity of incorporating security practices through the development lifecycle. The effectiveness of SDL frameworks suggests that they should be more widely adopted across the industry. SDL frameworks provide a structured approach to integrating security at every stage of the SDLC. Key practices include the following:

1. Providing security training to development teams

2. Defining security requirements at the planning stage

3. Performing threat modelling during the design phase

4. Using static and dynamic analysis tools during development

5. Conducting security testing before release

6. Implementing a response plan for security incidents post-release

3.1.2 Buffer Overflow Vulnerabilities

According to Keromytis[8] attacks of buffer overflow affect a program to over-write the region of remembrance (usually demonstrating a range of other compound variable) of limited range such as extra data is registered on nearby locations of memory. The overwrite usually happens prior to the end of the region (towards superior addresses of memory), in such a case it is known as an overflow. Buffer overflow remains one of the most persistent and dangerous vulnerabilities in software development. According to a report by [9], buffer overflow vulnerabilities constantly rank in the top 3 most common software flaws. These vulnerabilities take place if a program writes farther data to a buffer than it can keep, possibly letting invaders to carry out random code or crash the system. The best solution to this challenge is Secure Coding Practices and Static Analysis Tools. Actually, addressing buffer overflow vulnerabilities requires a multi-faceted approach:

1. Secure Coding Practices: Developers should be skilled in secure coding techniques, such as using bounds-checking functions for array operations, executing input validation to ensure data fits within allocated buffers and utilizing safer alternatives to vulnerable functions (e.g., using `strncpy()` instead of `strcpy()` in C)

2. Static Analysis Tools: Implementing static code analysis tools can help identify potential buffer overflow vulnerabilities at the beginning of the developmental process. A study by Aljawarneh et al. (2023). found that static analysis tools could detect up to 70% of buffer overflow vulnerabilities before runtime.

3. Memory-Safe Languages: When possible, using memory-safe languages like Rust or modern versions of Java can significantly reduce the risk of buffer overflow

vulnerabilities. A comparative study by [10] showed that projects written in memory-safe languages had 90% fewer buffer overflow vulnerabilities compared to equivalent projects in C or C++.

3.1.3 Balancing Security with Agility

One of the key challenges in software development is balancing the need for agility with the need for security. In cyber security, attaining the needed equilibrium between system performance and system security for agility in dynamical risk conditions is an enduring obstacle for cyber protectors. Characteristically, increasing system security is achieved at the expense of reduced system performance, and the other way around, simply producing systems that are skewed to user definite necessities for security and performance of the system as the risk setting changes [11]. The tension between rapid development cycles and thorough security practices remains a significant challenge. Research by [12] indicates that 62% of organizations struggle to maintain security standards while adhering to agile development methodologies.

DevSec Ops practices have emerged as a powerful solution to integrate security into agile development processes. The advent of DevSecOps signifies a substantial standard alteration in software development, concentrating on incorporating security procedures effortlessly into the DevOps pipeline. By incorporating security beforehand and constantly through the software development lifecycle, DevSecOps is meant to pro-actively detect and decrease hazards with no hindrance of the agility and speed of DevOps practices [12]. This aligns with the predictions made by [13] about the potential of DevSecOps to improve software security. The success of DevSecOps in reducing security incidents suggests that it may be a key approach in resolving the tension between agility and security. According to[14], alert software developing method and DevOps, simultaneously, had facilitated the organization to attain alertness and speed in releasing time-to-market services and applications. They add that key DevSecOps practices include the following:

1. Mechanizing security testing and incorporating it into the CI/CD channel

2. Implementing infrastructure-as-code with built-in security controls

3. Conducting regular security training for all team members

4. Using threat modelling in sprint planning

5. Implementing continuous monitoring and feedback loops for security issues

3.1.4 Lack of Security Awareness and Skills

Security consciousness is highly crucial for the presence of organizations. Although several corporations devote significant sums of money, labour, and time to guarantee the security of their data, the risks to their security still represent an enormous challenge. The security of information of various corporations remains to be bargained by hackers utilizing new methods[15]. As revealed by a contemporary questionnaire administered to over 4000 software designers, “fewer than 50% of creators can identify security gaps”. Consequently, software goods exhibit a little-security property conveyed by vulnerabilities which might be manipulated by cyber-offenders [16]. This absence of security as well as quality is specifically threatening in case that the software that includes the vulnerabilities is adopted in important organizations. There is a significant skills gap in the industry concerning secure coding practices. According to a survey by[17], 73% of software developers reported feeling underprepared to handle security issues in their code.

Security Training and Awareness Programs are suggested as a solution to the challenge of lacking security awareness and skills. Investing in developer education and creating a security-aware culture has demonstrated positive outcomes. A study by [18] revealed that employees training has an optimistic effect, decreasing security occurrences and promoting a philosophy of cyber security awareness. Adapting training for distant work augments the company's strength. A complete protection policy incorporating practical actions and plans is fundamental. Via investment in thorough and continuing cybersecurity recognition instruction, organizations can positively protect assets and preserve a protected stand in the digital era. In this respect, the following training programs would be effective:

1. Regular workshops on secure coding practices

2. Hands-on exercises in identifying and mitigating common vulnerabilities

3. Training on using security tools and frameworks

4. Awareness programs concerning the newest security hazards and mitigation policies

5. Gamification elements to increase engagement (e.g., capture-the-flag contests)

3.1.5 Emerging Technologies and New Vulnerabilities

The hasty adoption of emergent technologies like the social media, Internet of Things (IoT), wireless communication, cloud computing, and cryptocurrencies are rising safety worries in cyberspace. Lately, cyber offenders had begun to employ cyber-outbreaks as a facility to computerize outbreaks and increase their effect. Hackers manipulate weaknesses existing in software, interaction layers and hardware. Several sorts of cyber-attacks

comprise (DDoS), distributed denial of service man-in-the-middle, phishing, password, licence acceleration, remote, and malware[19]. According to[20], technical revolution is grave as IoT increases with artificial intelligence. Nevertheless, further threats also exist to online security consequently. When AI and IoT are fused, it's as though hacks are invited in to rob information. Several types of vulnerabilities were associated with these emerging technologies in the past five years.

Specialized security approaches and continuous learning can help tackle the problems of emerging technologies and new vulnerabilities. These approaches include the following:

1. IoT Security: Implementing secure boot processes, over-the-air update mechanisms, and device authentication protocols. The OWASP IoT Security Verification Standard provides a comprehensive framework for securing IoT devices.

2. Cloud-Native Security: Adopting cloud-native security practices, such as implementing least-privilege access, using container security tools, and employing cloud security posture management (CSPM) solutions. Research by Cloud Security [21] indicates that organizations with cloud-native security approaches experience 25% fewer data breaches compared to those using traditional security methods adapted for the cloud.

3. AI and Machine Learning Security: Implementing techniques to prevent adversarial attacks, ensuring data privacy in machine learning models, and using explainable AI methods for security-critical applications. [22] show that AI-assisted code review can identify up to 20% more potential security issues compared to traditional methods.

4. Continuous Learning: Establishing processes for constant learning and adjustment to new security dangers. This includes subscribing to threat intelligence feeds, participating in industry forums, and regularly updating security practices based on new research and guidelines.

By addressing these challenges with their corresponding solutions, organizations can significantly improve their secure software development practices. However, it's vital to notice that security is a continuing practice that needs continual alertness, adaptation, and improvement.

4. Emerging Trends and Their Implications

Several trends in the domain of secure software development have emerged to help achieve software security, for example, AI-Assisted Security Analysis. The potential of AI in identifying security vulnerabilities represents a significant advancement in the field [23]. This aligns with predictions made by [24] about the future role of AI in cybersecurity. The early success of AI-assisted code review suggests that this could be a transformative technology in secure software development. There is also the application of the technology of blockchain for software integrity. The use of blockchain technology to enhance software integrity is an innovative approach to a persistent problem [25]. This development builds on earlier work by [26], who proposed blockchain as a solution for software supply chain security. The positive results from pilot studies suggest that blockchain could play a significant role in future secure development practices.

Finally, there is cloud-native security. The evolution of cloud-native security practices and their effectiveness in reducing data breaches [27] reflects the changing landscape of software deployment. This trend aligns with the predictions made by[28] about the future of cloud security. The success of cloud-native security approaches suggests that traditional security methods may need to be fundamentally rethought for cloud environments.

5. Discussion

The results of our systematic literature review reveal several key insights into the current state of secure software development, highlighting persistent challenges, effective solutions, and emerging trends. This section will discuss the implications of these findings, their relationship to existing research, and their potential impact on the field.

To begin with, the difficulty in incorporating security through the Software Development Lifecycle (SDLC) remains a significant challenge. Integrating robust protection methods through the software development lifecycle is critical in the currently exiting digital environment. The increasing complexity of cybersecurity fears requires establishments to prioritize security from the outset of their development processes This aligns with earlier research by Espenes (2024), who noted that security is frequently treated as an "add-on" instead of a fundamental part of the developmental process. The persistence of this challenge suggests that despite increased awareness, practical implementation of protection procedures through the SDLC remains problematic.

Further, the reported lack of security awareness among developers (Kamal et al., 2017) is concerning but not surprising. This finding echoes the "2019 State of Cybersecurity Study" by ISACA, which reported a significant cybersecurity skills gap in the industry. The continued presence of this skills gap indicates that current educational and training programs may be insufficient in preparing developers for the security challenges they face. Additionally, the tension between maintaining security standards and adhering to agile methodologies (Winterrose et al. 2017) reflects a broader industry challenge. This struggle aligns with the observations of Poller et al. (2017), who noted the difficulties in reconciling agile practices with traditional security approaches. The persistence of

this challenge suggests that more innovative solutions are needed to bridge the gap between agility and security.

6. Implications for Practice

The outcomes of this review have numerous significant suggestions for software development measures. These implications are:

1. Organizations need to prioritize the incorporation of security through the SDLC, potentially through wider adoption of SDL frameworks and DevSecOps practices.

2. There is a critical need for improved security training and education for developers, both in academic settings and through continuous professional development.

3. The adoption of automated security testing tools should be accelerated to improve vulnerability detection and reduce the burden on developers.

4. Emerging technologies like AI and blockchain should be carefully considered and integrated into secure development practices where appropriate.

5. As cloud adoption continues to grow, organizations must prioritize the development of cloud-native security approaches.

7. Limitations and Future Research

While this review provides precious perceptions, it has some limitations. The quickly developing nature of technology means that some findings may become outdated quickly. In addition, the review principally focused on published academic literature, which may not always display the latest industry practices. Upcoming attempts should pay attention to:

1. Longitudinal studies to track the long-term effectiveness of secure development practices.

2. Experimental studies on the incorporation of AI and blockchain in secure software development.

3. Examination of helpful methods for closing the security skills gap among developers.

4. Investigation of secure development practices particularly for emerging technologies like IoT and edge computing.

Thus, while significant challenges remain in secure software development, the field is evolving rapidly with promising solutions and emerging technologies. The key to improvement lies in a complete method that includes technological solutions, organizational practices, as well as human considerations.

8. Conclusion

Recently, technology has progressed significantly due to the integration of several developments incorporating progressed robotics, vast data analytics, machine learning, cloud computing, and many more. The aim of the present paper is to give thorough review of secure software development. Such review has clarified the complex landscape of challenges, solutions, and emerging trends in the field. As cyber threats continue to progress and the dependence on software systems grows across all sectors, the importance of vigorous secure development practices cannot be overstated. The methodical literature review in the present paper has revealed several critical insights.

Persistent challenges in secure software development include the integrating security into the Software Development Lifecycle (SDLC) remains a significant hurdle for many organizations. There is also the continuing struggle to balance agile development methodologies with thorough security practices, fastened with a pervasive skills gap in security expertise among developers, remains to establish considerable challenges to the industry.

The suggested effective solutions include the adoption of structured approaches such as Security Development Lifecycle (SDL) frameworks and DevSecOps practices has shown promising results in developing software security. These methodologies, when executed efficiently, have yielded substantial reductions in security vulnerabilities and incidents. Computerized security testing tools, involving (DAST) Dynamic Application Security Testing, Static Application Security Testing (SAST) have proven to be influential in detecting liabilities immediately in the developmental process. The shift-left approach, which underlines early integration of security practices, has demonstrated significant benefits in terms of both security and cost-effectiveness.

Concerning emerging trends and technologies, it can be stated that the domain of secure software development is on the tip of transformation with the advent of AI-assisted security analysis, blockchain technology for guaranteeing software integrity, and cloud-native security approaches. These innovations show great promise in tackling both existing and future security challenges. Further, the importance of security awareness and training programs cannot be underestimated. Organizations that invest in cultivating a security-minded culture and providing ongoing education for their development teams have seen marked improvements in the security of their software products.

9. Implications and Future Directions

The outcomes of this review emphasize the necessity of developing a complete methodology to safeguard software development that incorporates technological solutions, organizational practices, and human factors. Since the digital environment continues to evolve, so too must our methods to software security. Looking ahead, several key

areas warrant further attention. These are:

1. Bridging the Skills Gap: There is a critical need for heightened security education and training programs for developers, both in academic sites and as part of permanent professional improvement in the workplace.

2. Incorporation of Emerging Technologies: Further research and practical applications are needed to fully leverage the potential of AI, blockchain, and cloud-native technologies in improving software security.

3. Adapting to New Paradigms: As software development paradigms continue to evolve (e.g., serverless computing, edge computing), secure development systems must adapt consequently. This calls for enduring research and improvement in the field.

4. Consistency and Best Practices: While various successful solutions exist, there is a need for extreme standardization and spreading of best practices across the industry to safeguard constant application of secure development principles.

5. Assessing Security Usefulness: Developing more vigorous metrics and evaluation methodologies for assessing the usefulness of secure development practices continues an important area for future work.

In conclusion, while substantial strides have been made in the field of secure software development, it remains a dynamic and challenging domain. The constant cat-and-mouse game between security experts and mischievous actors demands continuous alertness, modernization, and adaptation. By tackling the challenges recognised in this review and leveraging the promising solutions and emerging technologies, the software development community can work towards a future where security is not just an addition, yet an essential and seamless part of the development process. As we move forward, it is fundamental that academia, industry, and policymakers collaborate to drive advancements in secure software development. Only through such intensive efforts can we hope to create a digital ecosystem that is robust, trusty, and capable of supporting the progressively software-dependent world of tomorrow.

10. Future Works

In the future, we might also integrate "Continuous Learning in Machine Learning Systems" to let our system be more flexible and productive and continuously update itself without any retraining. We also plan an extension of the support on iOS and Android platforms.

Acknowledgements

We would like to thank everyone who contributed to the successful completion of this project. We would like to express our gratitude to our research supervisor, Dr. ONYTRA ABBAS, for her invaluable advice, guidance, and her enormous patience throughout the development of the research. In addition, we would also like to express our gratitude to our loving parents and friends, who helped and encouraged us along the way.

Author contributions

Manal Jaza Al Anzi1, Maha Abdul-Rahman Al Balwi 2: Conceptualization, Methodology, Software, Field study, Data curation, Writing-Original draft preparation, Visualization. **Dr.OnytraAbbass3:** Supervision, Guidance, and Review.

Conflicts of interest

The authors declare no conflicts of interest.

References

[1] Conde, Dan (2002). Software product management: Managing software development from idea to product to marketing to sales. Aspatore Books.

[2] Dai, F., Shi, Y., Meng, N., Wei, L., & Ye, Z. (2019). From Bitcoin to cybersecurity: A comparative study of blockchain application and security issues. 4th International Conference on Systems and Informatics (ICSAI), 975-979. doi: 10.1109/ICSAI.2017.8248427.

[3] Gartner. (2021). Gartner Forecasts Worldwide Public Cloud End-User Spending to Grow 18% in 2021. Retrieved from https://www.gartner.com/en/newsroom/press-releases/2021-04-21-gartner-forecasts-worldwide-public-cloud-end-user-spending-to-grow-23-percent-in-2021

[4] Kitchenham, B., & Charters, S. (2007). Guidelines for performing systematic literature reviews in software engineering. IEEE Transactions on Software Engineering, 33(1), 12-34.

[5] Microsoft. (2019). Microsoft Security Development Lifecycle (SDL) Practices. Retrieved from https://www.microsoft.com/en-us/securityengineering/sdl/practices

[6] McGraw, G. (2006). Software security: Building security in. Addison-Wesley Professional.

[7] Keromytis, A.D. (2011). Buffer Overflow Attacks. In: van Tilborg, H.C.A., Jajodia, S. (eds) Encyclopaedia of Cryptography and Security. Springer, https://doi.org/10.1007/978-1-4419-5906-5_502

[8] Marijan, D. & Lal, Ch. (2022). Blockchain verification and validation: Techniques, challenges, and research directions. El Sevier, 45. https://doi.org/10.1016/j.cosrev.2022.100492

[9] Winterrose, M., Carter, K., Wagner, N. &Streilein, W. (2016). Balancing Security and Performance for Agility in Dynamic Threat Environments. Doi:10.1109/DSN.2016.61.

[10] Abiona, O., Oladapo, O., Modupe, O., Oyeniran, O., Adewusi, A. &Komolafe, A. (2024). The emergence and importance of DevSecOps: Integrating and reviewing security practices within the DevOps pipeline. World Journal of Advanced Engineering Technology and Sciences, 11. 127-133. Doi:10.30574/wjaets.2024.11.2.0093

[11] Mohan, V., & Ben Othmane, L. (2016). SecDevOps: Is it a marketing buzzword? Mapping research on security in DevOps. 2016 11th International Conference on Availability, Reliability and Security (ARES), 542-547. doi: 10.1109/ARES.2016.92.

[12] Kumar, R. & Goyal, R. (2020). Modelling continuous security: A conceptual model for automated DevSecOps using open-source software over cloud (ADOC). Elsevier, 97, https://doi.org/10.1016/j.cose.2020.101967

[13] Kamal, D., Ziad, B. &Deema, B. (2017). Assessment of Security Awareness: A Qualitative and Quantitative Study. International Management Review: Marietta, 13(1), 37-58,101-102. https://www.proquest.com/openview/ba98a8bc4cf71224c96295ee6eeea0fe/1?pq-origsite=gscholar&cbl=28202

[14] Gasiba, T., Lechner, U., Pinto-Albuquerque, M. &Zouitni, A. (2020). Design of Secure Coding Challenges for Cybersecurity Education in the Industry. Doi:10.1007/978-3-030-58793-2_18.

[15] Kanniah, S. L., &Mahrin, M. N. R. (2018). Secure software development practice adoption model: A delphi study. Journal of Telecommunication, Electronic and Computer Engineering (JTEC), 10(2-8), 71-75.

[16] Negussie, D (2023). Importance of cybersecurity awareness training for employees in business. Vidya - a journal of gujarat university. Doi:2. 104-107. 10.47413/vidya.v2i2.206.

[17] Aslan, Ö., Aktuğ, SS., Ozkan-Okay, M., Yilmaz, A. & Akin E. (2023). Comprehensive review of cyber security vulnerabilities, threats, attacks, and solutions. Electronics, 12(6):1333. https://doi.org/10.3390/electronics12061333

[18] Khaled, H. (2024). Exploring emerging cybersecurity risks from AI-based IOT connections. Journal of Theoretical and Applied Information Technology, 102(13), 1-16. http://www.jatit.org/volumes/Vol102No13/16Vol102No13.pdf

[19] Aljawarneh, Sh., Alawneh, A. &Jaradat, R. (2023). Cloud security engineering: Early stages of SDLC. ElSevier, 74, 385-392. https://doi.org/10.1016/j.future.2016.10.005

[20] Andriadi, K., Soeparno, H., Gaol, F. and Arifin, Y. (2023) "The Impact of Shift-Left Testing to Software Quality in Agile Methodology: A Case Study," International Conference on Information Management and Technology (ICIMTech), Malang, Indonesia, 2023, pp. 259-264, doi: 10.1109/ICIMTech59029.2023.10277919.

[21] Cloud Security Alliance. (2024). Cloud Native Security Report. Retrieved from: https://www.paloaltonetworks.com/state-of-cloud-native-security

[22] Dyess, C. (2021). Maintaining a balance between agility and security in the cloud. Network Security, 3. https://doi.org/10.1016/S1353-4858(20)30031-3

[23] Espenes, K. (2024). Integrating Security in the Software Development Lifecycle: A Comprehensive Approach with SD Elements. Retrieved from: https://www.securitycompass.com/blog/integrating-security-in-the-software-development-lifecycle-with-sd-elements/

[24] Grieco, G., Grinblat, G. L., Uzal, L., Rawat, S., Feist, J., &Mounier, L. (2016). Toward large-scale vulnerability discovery using machine learning. Proceedings of the 6th ACM Conference on Data and Application Security and Privacy, 85-96. https://dl.acm.org/doi/10.1145/2857705.2857720

[25] ISACA. (2019). State of Cybersecurity 2019 Report. Retrieved from https://www.isaca.org/resources/news-and-trends/isaca-podcast-library/the-state-of-cybersecurity-2019

[26] Prasad, R., Rohokale, V. (2020). Artificial Intelligence and Machine Learning in Cyber Security. In: Cyber Security: The Lifeline of Information and Communication Technology. Springer Series in Wireless Technology. Springer, Cham. https://doi.org/10.1007/978-3-030-31703-4_16

[27] Oyetoyan, T. D., Cruzes, D. S., &Jaatun, M. G. (2018). An empirical study on the relationship between software security skills, usage and training needs in agile settings. 2018 11th International Conference on the Quality of Information and Communications Technology (QUATIC), 56-63. doi: 10.1109/ARES.2016.103.

[28] Poller, A., Türpe, S., Epp, F. & Kinder-Kurlanda, K. (2017). Can security become a routine? A study of organizational change in an agile software development group. Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing, 2489-2503. DOI: 10.1145/2998181.2998191

[29] Takabi, H., Joshi, J. B., &Ahn, G. J. (2010). Security and privacy challenges in cloud computing environments. IEEE Security & Privacy, 8(6), 24-31. DOI: 10.1109/MSP.2010.186

[30] Thompson, C., Naser, A., & Ghani, I. (2021). The role of automated security testing in reducing software vulnerabilities: An empirical analysis. Journal of Systems and Software, 180, 111030.