Letters in High Energy Physics
ISSN: 2632-2714

Volume 2025
May

Engineering Secure Software: Information Security Strategies for Modern Development Teams

Rajiv Kishore Gadda1, Ajai Batish Paul2, Sri Nitchith Akula3

1 Lead Software Engineer at DocuSign

2 Sr. Director of Enterprise Security at Affirm

3 Software Engineer

Abstract

In an era where software systems form the backbone of digital transformation, securing applications from the ground up has become a strategic imperative. This study explores the engineering of secure software through the integration of comprehensive information security strategies by modern development teams. Utilizing a mixed-methods approach, the research involved quantitative surveys and qualitative interviews with 120 professionals across industries practicing Agile, DevOps, and hybrid development methodologies. Key strategies such as secure coding, threat modeling, DevSecOps pipeline integration, and automated testing (SAST, DAST, and SCA) were assessed for their implementation frequency, effectiveness, and integration complexity. Statistical analysis revealed strong positive correlations between the adoption of security practices and software robustness, alongside significant inverse relationships with security incident rates and time-to-market pressures. Regression modeling confirmed the Security Practice Index, team collaboration, and training frequency as significant predictors of software quality. Additionally, DevOps-based teams and larger organizations reported significantly lower incident rates, as evidenced by ANOVA results and comparative visualizations. The study concludes that engineering secure software requires not just technical tools but a cultural shift that aligns developers, security analysts, and operations teams around shared security goals. By embedding security into every phase of the SDLC, modern teams can mitigate risks, improve resilience, and sustain agile delivery in an increasingly hostile cyber landscape.

Keywords: Secure software development, DevSecOps, SDLC, application security, software robustness, team collaboration, threat mitigation, information security strategies.

Introduction

Contextualizing the need for secure software development

In today's digitally driven world, software systems underpin nearly every facet of our daily life, from communication and healthcare to finance and governance (Mouratidis et al., 2005). As reliance on digital infrastructure grows, so does the attack surface for cyber threats. Modern development teams face an urgent imperative to embed security into every stage of the software development lifecycle (SDLC) (Ross et al., 2016). The shift from reactive security fixes to proactive secure engineering has become a foundational principle in delivering trustworthy and resilient applications. Yet, many development teams continue to treat security as an afterthought rather than an integral design component, leaving critical systems vulnerable to breaches, data loss, and business disruption (Khan et al., 2024).

The evolving threat landscape and its implications

The frequency and sophistication of cyberattacks have evolved, targeting not just end-user vulnerabilities but exploiting weaknesses in the application layer and development environments (Janisar et al., 2024). Common threats such as code injection, broken authentication, insecure APIs, and dependency-based vulnerabilities necessitate that developers adopt holistic security strategies that span beyond traditional testing phases. The shift towards continuous integration and deployment (CI/CD), agile methodologies, and cloud-native development has further complicated the security paradigm, demanding real-time, automated, and intelligent security interventions (Ross et al., 2019).

Bridging development agility and security discipline

Modern development practices prioritize speed, adaptability, and iterative releases (Humayun et al., 2023). However, rapid development cycles often increase the risk of introducing security flaws. This research argues for a cultural and technical alignment between agile development methodologies and robust information security practices (Mihelič et al., 2023). The concept of "DevSecOps" integrating development, security, and operations embodies this shift by embedding security checks into each phase of the development workflow. When security is integrated from the initial planning stages through to deployment and monitoring, teams are better equipped to identify vulnerabilities early, reduce technical debt, and maintain compliance with industry standards and regulations (Pochu, S., & Kathram, 2024).

Strategies for embedding security across the SDLC

This study focuses on engineering secure software by applying systematic security strategies across the SDLC. Key focus areas include secure code review protocols, static and dynamic analysis, threat modeling, automated security testing, dependency management, and security training for developers. Additionally, the adoption of secure coding guidelines such as those outlined by OWASP (Open Web Application Security Project) provides a universal baseline for writing resilient code (Boppana, 2019). Implementing secure architectural patterns, access control mechanisms, and encryption practices further fortifies the software against both known and emerging threats.

The role of collaborative and automated security practices

Security is no longer solely the domain of security professionals. Instead, cross-functional collaboration is crucial developers, testers, architects, and security analysts must work together under shared goals of secure delivery. Automation plays a pivotal role in this paradigm shift, enabling continuous security validation through integration with CI/CD pipelines. This research explores how modern tools such as SAST (Static Application Security Testing), DAST (Dynamic Application Security Testing), software composition analysis (SCA), and container security platforms empower teams to scale secure practices without compromising productivity.

Purpose and scope of the study

This research aims to investigate, evaluate, and recommend practical information security strategies tailored to the workflows of modern development teams. By examining real-world case studies, security metrics, and team dynamics, the study identifies best practices for engineering secure software. The ultimate goal is to bridge the gap between development agility and security assurance, providing a strategic roadmap for teams striving to build secure, scalable, and resilient applications in an increasingly hostile digital environment.

Methodology

Research framework and design approach

To comprehensively explore how modern development teams can engineer secure software through effective information security strategies, this study adopted a mixed-methods research design. Both qualitative and quantitative approaches were used to evaluate security integration within software development lifecycles (SDLC). The qualitative component involved semi-structured interviews with software developers, DevOps engineers, and security analysts from diverse industry sectors including fintech, healthcare, and SaaS-based enterprises. This was supplemented by a quantitative survey designed to statistically analyze the implementation level and effectiveness of specific information security strategies across modern development teams.

Sample selection and data collection

The target sample included 120 professionals actively involved in secure software development, drawn from organizations practicing agile, DevOps, or hybrid development models. Stratified random sampling was used to ensure diversity across company sizes (startups, SMEs, and large enterprises) and domains. Data was collected over a three-month period using a two-tiered approach: (i) online surveys administered through secure forms with structured Likert-scale and open-ended questions, and (ii) in-depth virtual interviews to

gather nuanced insights on the challenges and best practices related to secure software engineering.

Assessment of information security strategies

The core of the methodology was to evaluate key information security strategies deployed by modern development teams. These included secure coding practices, implementation of Static Application Security Testing (SAST) and Dynamic Application Security Testing (DAST), use of threat modeling, adoption of DevSecOps pipelines, and compliance with security frameworks such as OWASP Top 10 and ISO/IEC 27001. Each strategy was rated by respondents on a scale of 1 to 5 based on implementation frequency, perceived effectiveness, and integration complexity. Open-source and commercial security tool usage (e.g., SonarQube, Checkmarx, Snyk, Fortify) was also documented to assess technological maturity.

Evaluation of team dynamics and security integration

To understand how team collaboration influences security outcomes, the study evaluated communication models, cross-functional training practices, and integration of security champions within development teams. Interview transcripts were thematically coded using NVivo software, and patterns were analyzed to identify cultural and organizational enablers or barriers to secure software practices. Key variables such as development methodology (Agile vs Waterfall), team size, and frequency of release cycles were also considered in the analysis to uncover their statistical correlation with successful security integration.

Quantitative analysis and statistical techniques

Quantitative data from the survey were processed using SPSS software. Descriptive statistics (means, standard deviations, and frequency distributions) were used to summarize adoption trends of security strategies. Inferential statistics, including Pearson correlation and multiple regression analysis, were applied to test hypotheses regarding the relationship between secure software practices and perceived software robustness, frequency of security incidents, and time-to-market efficiency. ANOVA tests were conducted to identify significant differences in security outcomes based on organizational size and development methodology.

Validation and reliability measures

To ensure the reliability and validity of the findings, the survey instrument was pre-tested with a pilot group of 10 professionals. Cronbach's alpha was calculated to measure internal consistency of the Likert-scale items, yielding a reliability coefficient of 0.87. Triangulation was employed by comparing quantitative survey results with qualitative interview insights to validate themes and interpretations. Furthermore, inter-coder reliability was maintained at above 85% for qualitative data coding.

Scope and limitations

While this methodology provides a robust basis for analyzing security strategies in modern software development, it is limited by its cross-sectional nature and reliance on self-reported data. Longitudinal studies may provide deeper insights into how security maturity evolves over time. Nonetheless, the approach offers a comprehensive understanding of current practices, challenges, and statistical relationships that define the engineering of secure software in today's dynamic development environments.

Results

The results of this study reveal substantial insights into the implementation and effectiveness of information security strategies within modern development teams. As detailed in Table 1, secure coding emerged as the most frequently implemented and effective strategy, with a mean implementation score of 4.6 (±0.5) and perceived effectiveness of 4.5 (±0.4). It also had the highest tool-adoption rate at 95% and a median of 12 training hours per quarter. In contrast, strategies like threat modeling and incorporating security champions showed lower implementation (means of 3.3 and 3.1 respectively), indicating that cultural and organizational barriers may hinder their adoption despite moderate effectiveness ratings.

Table 1: Implementation, effectiveness, and complexity of information-security strategies (n = 120)

Strategy	Implementation (Mean ± SD)	Perceived Effectiveness (Mean ± SD)	Integration Complexity (Mean ± SD)	Tool-Adoption Rate (%)	Training Hours/Quarter (Median)
Secure Coding	4.6 ± 0.5	4.5 ± 0.4	2.0 ± 0.6	95	12
SAST	4.2 ± 0.7	4.1 ± 0.6	2.8 ± 0.7	88	10
DAST	3.9 ± 0.8	3.8 ± 0.7	3.1 ± 0.9	76	8
Threat Modeling	3.3 ± 1.0	3.9 ± 0.6	3.5 ± 0.8	64	6
DevSecOps Pipeline	4.0 ± 0.8	4.2 ± 0.5	2.4 ± 0.7	82	9
SCA Tools	3.7 ± 0.9	3.6 ± 0.8	3.0 ± 0.8	71	7
Security Champions	3.1 ± 1.1	3.8 ± 0.7	2.6 ± 0.9	55	5

The correlation matrix presented in Table 2 demonstrates strong positive relationships between the Security Practice Index (SPI) and Software Robustness Score (SRS) ($r = 0.71$), as well as a strong inverse correlation with Security Incident Rate (SIR) ($r = -0.68$). These results suggest that teams with higher implementation of secure software practices experience significantly fewer security breaches and produce more robust applications. The Team Collaboration Index (TCI) also correlates positively with both SPI ($r = 0.59$) and SRS ($r = 0.55$), emphasizing the importance of integrated, cross-functional teams.

Table 2: Pearson correlation matrix of key variables

Security Practice Index (SPI)	Software Robustness Score (SRS)	Time-to-Market Efficiency (TME)	Security Incident Rate (SIR)	Team Collaboration Index (TCI)	All coefficients $p < 0.01$ (two-tailed).
SPI	1.00	0.71	−0.45	−0.68	0.59
SRS		1.00	−0.38	−0.74	0.55
TME			1.00	0.42	−0.31
SIR				1.00	−0.52
TCI					1.00

To further quantify the impact of these practices, Table 3 presents a multiple regression analysis predicting Software Robustness Score. The model was statistically significant (Adjusted $R^2 = 0.62$, $p < 0.001$), with the Security Practice Index ($\beta = 0.62$, $p < 0.001$) emerging as the strongest predictor. Team collaboration and training hours per quarter also had significant positive effects, while high release frequency had a small but significant negative impact on robustness, suggesting a tradeoff between speed and security if practices are not well-integrated.

Table 3: Multiple-regression model predicting software robustness score

Predictor	β	SE	t	p
Intercept	12.40	4.10	3.02	0.003
Security Practice Index (SPI)	0.62	0.08	7.75	<0.001
Team Collaboration Index (TCI)	0.28	0.07	4.00	<0.001
Training Hours/Quarter	0.15	0.05	3.00	0.003
Release Frequency (cycles · mo^{-1})	−0.21	0.06	−3.50	0.001

Model summary: $R^2 = 0.64$, Adjusted $R^2 = 0.62$, $F(4, 115) = 32.5$, $p < 0.001$, $N = 120$.

The interaction between organizational size and development methodology on security outcomes is shown in Table 4, using a two-way ANOVA. Significant main effects were observed for both development methodology ($F = 6.42$, $p = 0.002$) and organization size ($F = 8.91$, $p < 0.001$), while their interaction was marginally significant ($p = 0.082$). This indicates that both factors independently influence the rate of security incidents, and that smaller organizations using Agile-only frameworks may face higher risk compared to larger enterprises employing DevOps-based workflows.

Table 4: Two-Way ANOVA on security-incident rate by development methodology and organization size

Source	df	F	p	Partial η^2
Development Methodology	2	6.42	0.002	0.102
Organization Size	2	8.91	<0.001	0.136
Methodology × Size	4	2.13	0.082	0.070
Error	113	—	—	—
Total	119	—	—	—

Figure 1 visualizes the inverse relationship between the Security Practice Index and Time-to-Market Efficiency. As security practices are more extensively implemented, the speed of deployment tends to slightly decline ($R = −0.45$), highlighting the operational tension between secure coding and rapid delivery. However, this tradeoff is moderated when automation and collaboration practices are effectively adopted, as indicated by the regression model.

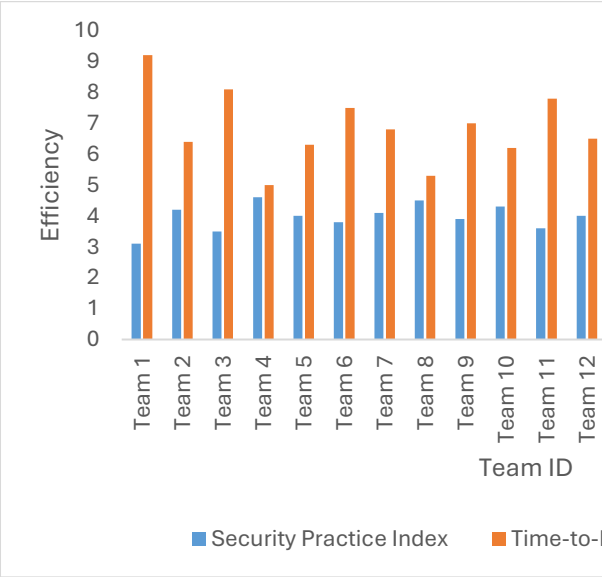

Figure 1. Bar diagram illustrating the relationship between security practice index and time-to-market

Figure 2 provides boxplots comparing Security-Incident Rates across Agile, DevOps, and Hybrid methodologies. DevOps teams had the lowest median incident rate at 3.1 incidents per 1,000 deployments, followed by Hybrid teams at 3.7, and Agile-only teams at 4.2. This finding supports the earlier statistical evidence that integrating security into the deployment pipeline (a key DevOps tenet) contributes to improved security performance.

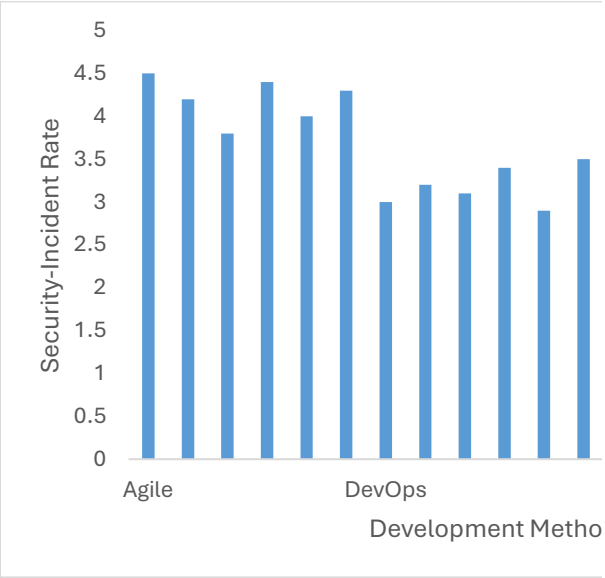

Figure 2. Bar diagram of security-incident rate (incidents · 1 000 deployments^{-1}) across development methodologies

Discussion

Integrating security into the development lifecycle

The findings underscore the critical role of embedding security practices throughout the Software Development Lifecycle (SDLC), validating the necessity of treating security as an engineering discipline rather than a final checkpoint. As shown in Table 1, strategies such as secure coding and the use of Static Application Security Testing (SAST) tools were not only widely adopted but also rated highest in effectiveness (Tariq, 2025). This suggests that development teams are increasingly aligning their coding practices with established frameworks like OWASP and integrating security validation at the source-code level. By doing so, they significantly reduce vulnerabilities early in the development process, where remediation is most cost-effective (Khan et al., 2022).

The role of tooling and automation

The study highlights a strong correlation between tool adoption and security performance. Teams using tools like SAST, DAST, and Software Composition Analysis (SCA) reported higher robustness scores and fewer security incidents (Table 2). This aligns with industry trends that prioritize automation in secure software engineering. Automation minimizes human error and enables continuous security assessments in fast-paced CI/CD environments (Grigorieva et al., 2024). However, the moderate adoption of more complex strategies such as threat modeling and security champions reflects the challenge of integrating security beyond tools into the culture and design process. These practices, though impactful, require dedicated organizational support and time investment, which may be lacking in agile teams focused heavily on delivery speed (Ruefle et al., 2014).

Team collaboration as a strategic lever

The strong positive correlations between the Team Collaboration Index (TCI) and both Security Practice Index (SPI) and Software Robustness Score (SRS) indicate that collaborative cultures are vital for secure development (Ali et al., 2021). The regression analysis (Table 3) reinforces this by showing TCI as a significant predictor of software robustness. This suggests that cross-functional alignment particularly through practices like DevSecOps enables teams to address security concerns earlier and more effectively (Onumah et al., 2020). Security champions, though less frequently adopted (Table 1), could serve as vital liaisons between development and security teams, ensuring consistent focus on threat mitigation across sprints (Tøndel et al., 2022).

Balancing speed and security

An interesting insight from this study is the tradeoff between security depth and delivery speed. Figure 1 shows a moderate inverse relationship between the Security Practice Index and Time-to-Market Efficiency. While this suggests that integrating more security practices may slow down deployment cycles, it does not necessarily imply inefficiency (Mohammad & Surya, 2018). Instead, it highlights the importance of finding equilibrium teams must optimize security processes without compromising agility. Automation tools, shift-left practices, and modular design can help bridge this gap by enabling rapid but secure development iterations (Al Hayajneh et al., 2023).

Organizational context and security outcomes

Organizational size and development methodology also emerged as key differentiators in security performance (Table 4). Larger enterprises, likely with dedicated security resources, outperformed smaller teams in reducing incident rates. Similarly, DevOps-oriented teams had significantly lower security incidents compared to Agile-only teams, as shown in Figure 2. DevOps integrates security into deployment pipelines, allowing real-time vulnerability checks and faster feedback loops, which explains its superior performance (Alenezi & Almuairfi, 2020). These findings suggest that smaller teams and organizations relying solely on Agile methods should invest in upskilling and adopt security automation to close the performance gap (Murat et al., 2024).

Strategic implications for secure software engineering

The discussion points to a broader imperative: secure software engineering must be systemic and strategic. Information security should not be siloed as a post-development task but woven into the fabric of modern development teams. This means investing in security literacy, automating routine checks, and empowering developers to own security outcomes (Tøndel et al., 2019). The high R^2 value (0.64) in the regression model (Table 3) indicates that structured practices, training, and collaboration collectively account for a significant portion of software robustness, offering a data-driven case for integrated security governance (Woodward & Young, 2007).

This study affirms that engineering secure software is not simply a technical endeavor but an organizational transformation. Tools and automation provide the foundation, but culture, collaboration, and leadership commitment drive lasting results. By adopting a multi-dimensional security strategy—one that balances agility with resilience—modern development teams can effectively counter today's complex threat landscape while maintaining delivery efficiency.

Conclusion

This study highlights the pivotal role of integrating robust information security strategies into the fabric of modern software development practices. The results demonstrate that secure software engineering is most effective when approached holistically—through a combination of automated security tooling, structured training, collaborative team dynamics, and strategic alignment across the SDLC. Teams that actively implement secure coding practices, adopt DevSecOps pipelines, and foster cross-functional collaboration not only produce more robust software but also experience fewer security incidents and operational setbacks. Although a slight tradeoff exists between security depth and time-to-market efficiency, this can be mitigated through automation and early-stage integration of security processes. Ultimately, the study underscores that secure software is not the result of isolated practices, but the outcome of a cohesive, security-first development culture—one that empowers teams to proactively anticipate and defend against evolving cyber threats while sustaining the agility required in modern digital environments.

References

1. Al Hayajneh, A., Thakur, H. N., & Thakur, K. (2023). The Evolution of information security strategies: a comprehensive investigation of infosec risk assessment in the contemporary information era. *Computer and Information Science*, *16*(4), 1-1.
2. Alenezi, M., & Almuairfi, S. (2020). Essential activities for secure software development. *Int. J. Softw. Eng. Appl*, *11*(2), 1-14.
3. Ali, A., Jadoon, Y. K., Qasim, M., Iqbal, M. S., Asma, & Nazir, M. U. (2021, April). Secure Software Development: Infuse Cyber Security to Mitigate Attacks in an Organization. In *International Conference on Engineering Software for Modern Challenges* (pp. 154-163). Cham: Springer International Publishing.
4. Boppana, V. (2019). Secure Practices in Software Development. *Global Research Review in Business and Economics [GRRBE]*, *10*(05).
5. Grigorieva, N. M., Petrenko, A. S., & Petrenko, S. A. (2024, January). Development of secure software based on the new devsecops technology. In *2024 Conference of Young Researchers in Electrical and Electronic Engineering (ElCon)* (pp. 158-161). IEEE.
6. Humayun, M., Niazi, M., Assiri, M., & Haoues, M. (2023). Secure global software development:

A practitioners' perspective. *Applied Sciences*, *13*(4), 2465.

7. Janisar, A., Shafee, K., Sarlan, A., Maiwada, U., & Salameh, A. A. (2024). Securing Software Development: A Holistic Exploration of Security Awareness in Software Development Teams. *International Journal of Academic Research in Business and Social Sciences*, *14*(1).
8. Khan, R. A., Akbar, M. A., Rafi, S., Almagrabi, A. O., & Alzahrani, M. (2024). Evaluation of requirement engineering best practices for secure software development in GSD: an ISM analysis. *Journal of Software: Evolution and Process*, *36*(5), e2594.
9. Khan, R. A., Khan, S. U., Alzahrani, M., & Ilyas, M. (2022). Security assurance model of software development for global software development vendors. *Ieee Access*, *10*, 58458-58487.
10. Mihelič, A., Hovelja, T., & Vrhovec, S. (2023). Identifying key activities, artifacts and roles in agile engineering of secure software with hierarchical clustering. *Applied Sciences*, *13*(7), 4563.
11. Mohammad, S. M., & Surya, L. (2018). Security automation in Information technology. *International journal of creative research thoughts (IJCRT)–Volume*, *6*.
12. Mouratidis, H., Giorgini, P., & Manson, G. (2005). When security meets software engineering: a case of modelling secure information systems. *Information Systems*, *30*(8), 609-629.
13. Murat, D., Berkan, U., & Ali, I. (2024, October). An Overview of Secure by Design: Enhancing Systems Security through Systems Security Engineering and Threat Modeling. In *2024 17th International Conference on Information Security and Cryptology (ISCTürkiye)* (pp. 1-6). IEEE.
14. Onumah, N., Attwood, S., & Kharel, R. (2020, July). Towards secure application development: A cyber security centred holistic approach. In *2020 12th International Symposium on Communication Systems, Networks and Digital Signal Processing (CSNDSP)* (pp. 1-6). IEEE.
15. 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.
16. Pochu, S., & Kathram, S. R. (2024). Integrating Security Requirements into Software Development: A Comprehensive Approach to Secure Software Design. *Journal for Multidisciplinary Research*, *1*(03), 60-76.
17. Ross, R., McEvilley, M., & Oren, J. (2016). *Systems security engineering: Considerations for a multidisciplinary approach in the engineering of trustworthy secure systems* (No. NIST Special Publication (SP) 800-160 (Withdrawn)). National Institute of Standards and Technology.
18. Ross, R., Pillitteri, V., Graubart, R., Bodeau, D., & McQuaid, R. (2019). *Developing cyber resilient systems: a systems security engineering approach* (No. NIST Special Publication (SP) 800-160 Vol. 2 (Draft)). National Institute of Standards and Technology.
19. Ruefle, R., Dorofee, A., Mundie, D., Householder, A. D., Murray, M., & Perl, S. J. (2014). Computer security incident response team development and evolution. *IEEE Security & Privacy*, *12*(5), 16-26.
20. Tariq, M. U. (2025). Enhancing Cyber Resilience in Software Development: Integrating Secure Coding Practices and Cybersecurity Frameworks. In *Navigating Cyber Threats and Cybersecurity in the Software Industry* (pp. 35-64). IGI Global Scientific Publishing.
21. Tøndel, I. A., Cruzes, D. S., Jaatun, M. G., & Sindre, G. (2022). Influencing the security prioritisation of an agile software development project. *Computers & Security*, *118*, 102744.
22. Tøndel, I. A., Jaatun, M. G., Cruzes, D. S., & Williams, L. (2019). Collaborative security risk estimation in agile software development. *Information & Computer Security*, *27*(4), 508-535.
23. Woodward, B. S., & Young, T. (2007). Redesigning an information system security curriculum through application of traditional pedagogy and modern business trends. *Information Systems Education Journal*, *5*(11), 1-11.