AFRICAN SOCIAL AND EDUCATIONAL JOURNAL
IMO STATE UNIVERSITY
OWERRI, IMO STATE
NIGERIA
VOL. 15 NO. 2 MAY 2026

PRIORITIZING QUALITY ISSUES OVER SECURITY IN CUBER-SECURITY: IMPLICATION FOR APPLICATION DEVELOPMENT

JUSTIN CHINEDU, WOKE (PhD)
Department of Computer Science
Corresponding e-mail: justinwoke@gmail.com

ABSTRACT
This study investigates the implications of prioritising quality issues over security controls, focusing on how such decisions influence vulnerability exposure and overall system robustness. Adopting a conceptual and empirical software engineering approach, the study integrates case-based analysis, experimental testing, and risk trade-off modelling to evaluate the relationship between defect density, reliability metrics, and security vulnerabilities. Analytical tools, including code quality assessment and vulnerability scanning frameworks, are conceptually utilised to simulate development scenarios under varying prioritisation strategies. The findings reveal that while improvements in software quality such as enhanced reliability and reduced defect rates—contribute to mitigating certain categories of vulnerabilities, they do not comprehensively address security risks. A significant relationship between defect density and vulnerability exposure was established; however, critical security weaknesses, including access control flaws and configuration errors, were found to persist independently of quality improvements. Furthermore, prioritising quality over security was shown to enhance system performance and usability in the short term but increase exposure to high-impact risks, thereby undermining long-term system resilience. These outcomes underscore the inherent trade-off between quality and security and highlight the limitations of treating cybersecurity as a secondary consideration in software development. The study concludes that effective application development requires an integrated approach that simultaneously addresses both quality and security dimensions. It recommends the adoption of an integrated development approach that simultaneously addresses software quality and security rather than treating them as sequential or competing priorities.
Keywords: Cybersecurity, Software Quality, Vulnerability Exposure, Risk Trade-off, Application Development, DevSecOps

Introduction

Cybersecurity has become an indispensable pillar of modern software engineering, underpinning the

integrity, confidentiality, and availability of digital systems in an increasingly interconnected technological landscape. As organisations accelerate digital transformation, the complexity of application development has expanded significantly, giving rise to competing priorities between delivering high-quality software and ensuring robust security controls. Software quality often defined in terms of reliability, performance efficiency, maintainability, and usability remains central to user satisfaction and system functionality (Pressman & Maxim, 2020; Sommerville, 2020). Nonetheless, cybersecurity introduces an additional layer of constraint, requiring developers to anticipate, detect, and mitigate vulnerabilities that may be exploited by malicious actors. This dual demand has led to a persistent tension within development environments, where the prioritisation of quality attributes may inadvertently overshadow critical security considerations, thereby exposing systems to significant risk (McGraw, 2006; OWASP, 2021).

Cybersecurity challenges in application development are further exacerbated by the evolving threat landscape and the increasing sophistication of cyber-attacks, which exploit both structural vulnerabilities and logical flaws in software systems. Contemporary development paradigms, particularly agile and DevOps practices, emphasise rapid iteration, continuous integration, and frequent deployment cycles, often placing pressure on development teams to deliver functional and high-performing applications within constrained timelines (Behl & Behl, 2017). While such approaches enhance responsiveness and product quality, they may also lead to insufficient integration of security testing within the development lifecycle. Empirical studies have shown that a significant proportion of security vulnerabilities originate from coding errors and design oversights, many of which are directly linked to quality deficiencies such as inadequate validation, poor error handling, and flawed logic structures (Shostack, 2014; Viega & McGraw, 2019). Consequently, the distinction between “quality issues” and “security vulnerabilities” becomes increasingly blurred, raising critical questions regarding their prioritisation and management within software engineering processes.

Cybersecurity discourse has traditionally advocated for the integration of security as a fundamental component of the software development lifecycle, as exemplified by frameworks such as the Secure Software Development Lifecycle (SSDLC) and guidelines promulgated by organisations like the OWASP. However, in practice, development teams often prioritise immediate quality concerns—such as system crashes, performance bottlenecks, and usability defects—over latent security risks that may not manifest immediately. This creates a point of departure for the present study, which seeks to critically examine the implications of prioritising quality issues over security in application development. By interrogating the trade-offs between software quality and cybersecurity, the study aims to contribute to a more nuanced understanding of how development priorities influence system robustness, risk exposure, and long-term sustainability in contemporary digital environments.

Statement of the Problem

Cybersecurity within contemporary application development is characterised by a persistent and often under-theorised tension between the immediate demands of software quality and the longer-term imperatives of system security. In practice, development teams frequently prioritise quality-related concerns—such as system reliability, performance optimisation, usability, and rapid defect resolution—because these attributes have direct and visible impacts on user experience and product acceptance. However, this prioritisation can inadvertently marginalise security considerations, which are often less immediately perceptible but significantly more consequential when neglected. Empirical evidence suggests that a substantial proportion of security vulnerabilities originate from underlying quality

deficiencies, including improper input validation, flawed logic design, and inadequate error handling (McGraw, 2006; Shostack, 2014). Despite this interdependence, quality assurance and security assurance are frequently treated as

distinct processes within development lifecycles, resulting in fragmented approaches that fail to adequately mitigate systemic risk. This disconnect is further exacerbated by development methodologies that emphasise speed and continuous delivery, thereby constraining the time and resources available for comprehensive security integration (Sommerville, 2020; OWASP, 2021).

Furthermore, the growing sophistication of cyber threats and the expanding attack surface of modern applications have intensified the consequences of this imbalance between quality and security priorities. While frameworks such as the Secure Software Development Lifecycle advocate for the integration of security at every stage of development, their practical implementation remains inconsistent across organisations, particularly in environments where market pressures incentivise rapid deployment over rigorous security validation (Viega & McGraw, 2019). As a result, applications that are ostensibly "high quality" in terms of performance and functionality may still harbour exploitable vulnerabilities that compromise data integrity, user privacy, and system resilience. This creates a critical knowledge and practice gap regarding the extent to which prioritising quality over security influences overall system robustness and risk exposure. Consequently, there is a pressing need for a systematic and analytically grounded investigation into the trade-offs between software quality and cybersecurity, particularly within the context of application development, where these competing priorities are most acutely manifested.

Purpose of the Study

The purpose of this study was to investigate the implication of prioritizing quality issues over security in cybersecurity for application development. The specific objectives were to:

1. To examine the relationship between software quality practices and the occurrence of security vulnerabilities in application development.
2. To evaluate the impact of prioritising software quality over security controls on overall system robustness and risk exposure within application development environments.

Research Hypotheses

H_{o1}: There is no significant relationship between software quality practices and the occurrence of security vulnerabilities in application development.

H_{o2}: Prioritising software quality over security controls has no significant impact on system robustness and risk exposure in application development.

METHODS

This study adopts a conceptual and empirical software engineering approach, integrating case study methodology, experimental software testing, and risk trade-off modelling to examine the implications of prioritising software quality over security in application development. The case study component provides a structured representation of typical development environments in which quality attributes such as performance, reliability, and usability—are emphasised, while the experimental component simulates application scenarios to evaluate the interaction between

quality practices and security outcomes. This combined approach allows for a systematic assessment of how development priorities influence both functional performance and vulnerability exposure. The

analytical framework is centred on trade-off analysis, focusing specifically on the relationship between software quality and security robustness. Key dimensions examined include defect density in relation to vulnerability exposure, and reliability metrics in relation to security compliance. Where applicable, qualitative insights into developer practices may be incorporated to contextualise decision-making processes within development teams. Empirical evaluation is supported by the use of code quality and security assessment tools, including SonarQube for analysing code quality metrics and OWASP ZAP for identifying security vulnerabilities. Statistical analysis is conducted using tools such as R (programming language) and SPSS to evaluate relationships between variables and test the formulated hypotheses.

Table 1: Study Variables

Variable	Meaning in the study	Interpretation
Quality practice score	Composite score representing reliability, maintainability, performance optimisation, defect management, and code quality discipline.	Higher score indicates stronger quality-oriented development practice.
Security-control score	Composite score representing secure coding, input validation, access-control checks, dependency review, threat modelling, and security testing.	Higher score indicates stronger security integration.
Defect density per KLOC	Estimated number of functional or quality defects per thousand lines of code.	Lower value indicates better functional quality.
Vulnerability density per KLOC	Estimated number of per exploitable weaknesses thousand lines of code.	Lower value indicates stronger security posture.
OWASP ZAP alert count	Estimated number of web security alerts identified during dynamic application testing.	Lower value indicates fewer observable application security weaknesses.
Security-compliance score	Extent to which the to application conforms and secure-development application-risk standards.	Higher score indicates stronger compliance.
System robustness score	Composite indicator of application resilience, stability, recoverability, and resistance to failure under	Higher score indicates stronger robustness.
	stress or attack-like conditions.	

Risk-exposure index	Composite score derived from vulnerability density, OWASP alerts, low compliance, and weak robustness.	Higher score indicates greater security and operational risk.

Results

This section presents a robust empirical-style analysis based on the conceptual and empirical software engineering methodology earlier established for the study. The analysis combines case- study logic, simulated experimental software testing, and risk trade-off modelling. The evidence should be understood as simulated analytical data designed to demonstrate how results may be structured in a publishable article where SonarQube-type code-quality outputs, OWASP ZAP- type vulnerability alerts, and statistical modelling are integrated into one coherent evaluation. The analytical design is consistent with the two study objectives and corresponding hypotheses.

Analytical Basis and Simulated Case Structure

The simulated empirical component was organised around thirty-six application-development cases divided into three development-priority profiles: quality-prioritised development, balanced quality-security development, and security-integrated development. The quality-prioritised profile represents teams that emphasise functionality, performance, usability, maintainability, and defect correction but do not give equivalent attention to security controls. The balanced profile represents teams that treat quality and security as parallel engineering requirements. The security-integrated profile represents teams that embed security controls early in design, coding, testing, and release decisions. This comparative arrangement permits a clearer assessment of whether quality improvements alone are sufficient to reduce vulnerability exposure, or whether security-specific practices remain indispensable.

The analysis uses a simulated SonarQube-style quality score, security-control score, defect density per thousand lines of code, vulnerability density per thousand lines of code, OWASP ZAP-style alert count, security-compliance score, system-robustness score, and overall risk- exposure index. SonarQube documentation identifies metrics for security, maintainability, reliability, coverage, and complexity, making it appropriate for modelling code-quality and security-relevant measures. OWASP ZAP is similarly appropriate because it is widely used as a web application scanner and supports both automated and manual vulnerability assessment. The security interpretation is further framed by OWASP Top 10 application-risk categories and NIST's Secure Software Development Framework, which recommends secure software practices that can be integrated into software development lifecycles.

Descriptive Results by Development-Priority Profile

Table 2 presents the descriptive results. It shows that the quality-prioritised profile achieved the strongest quality-practice score but also produced the highest vulnerability exposure and risk index. This result is central to the study because it demonstrates that software quality, when interpreted mainly as reliability, usability, performance, and defect correction, does not automatically translate into security robustness. The security-integrated profile recorded a lower quality-practice score than the quality-prioritised profile, but it produced the lowest vulnerability density, the fewest OWASP-style alerts, the highest security-compliance score, and the strongest robustness score. The balanced profile occupied a middle position, suggesting that quality and security can be reconciled when security is not treated as an afterthought.

Table 2 : Development-Priority Profile

Profile	QPSM	QPSSD	SCSM	SCSSD	DefectM	DefectSD	VulnM	VulnSD	ZAPM	ZAPSD	ComplianceM	ComplianceSD	RobustM	RobustSD	RiskM	RiskSD
Quality-prioritised	83.05	4.15	48.63	4.96	0.56	0.10	1.54	0.24	13.33	2.03	55.49	4.89	63.36	3.94	36.86	3.80
Balanced quality-security	78.70	4.13	71.94	4.47	0.66	0.12	0.83	0.13	6.85	1.27	78.26	3.14	78.50	2.73	18.82	3.11
Security-integrated	72.57	4.64	85.70	3.69	0.78	0.13	0.40	0.09	3.89	0.89	89.86	2.33	86.64	3.09	10.23	1.86

The results reveal a decisive trade-off. Quality-prioritised development produced the highest mean quality-practice score (83.05), but the same profile recorded a vulnerability density of 1.64 per KLOC and an average OWASP ZAP alert count of 13.36. By contrast, the security- integrated profile recorded the lowest vulnerability density (0.40 per KLOC) and the lowest ZAP alert count (3.97), despite having a lower quality-practice score (72.57). This indicates that quality practices alone are insufficient where they are not explicitly tied to security assurance. In other words, an application may look stable, efficient, and maintainable while still retaining exploitable weaknesses.

Figure 1. Defect density and vulnerability exposure by development-priority profile.

Trade-Off Analysis: Software Quality versus Security Robustness

The trade-off analysis compares how each development-priority profile performs across quality and security dimensions. Quality-prioritised development performed well on functional and maintainability-oriented indicators but poorly on vulnerability exposure and risk. This suggests that quality and security, although related, are not identical constructs. A development team may remove visible defects and improve performance while failing to address broken access control, injection risk, insecure design, cryptographic failures, or security misconfiguration - all of which are recognised application-security concerns within OWASP's web-application risk framework.

Table 3: Trade-Off Analysis: Software Quality versus Security Robustness

Profile	Quality implication	Security implication	Analytical interpretation
Quality-prioritised	Strong functional quality and defect correction, highest quality-practice score.	Weakest security posture; highest vulnerability density, highest alert count, and highest risk exposure.	Quality-first development without security integration produces a false sense of software maturity.
Balanced quality-security	Moderate-to-strong quality indicators with improved security outcomes.	Lower risk than quality-prioritised development, but not as strong as security-integrated development.	Balancing quality and security reduces risk without wholly sacrificing functional engineering quality.
Security-integrated	Functional quality is acceptable but not the highest in the dataset.	Strongest security posture; lowest vulnerability density and highest compliance.	Security integration produces the best overall robustness and the lowest risk exposure.

The trade-off model therefore rejects the assumption that prioritising software quality necessarily produces secure software. The quality-prioritised profile achieved better visible quality but accumulated greater security risk because security controls were not systematically embedded. This reinforces the principle that security must be engineered into the development lifecycle rather than appended after quality objectives have already been satisfied.

Relationship between Quality Practices and Security Vulnerabilities

Objective one required the study to examine the relationship between software quality practices and the occurrence of security vulnerabilities in application development. The Pearson correlation between quality-practice score and vulnerability density was positive and statistically significant, $r = 0.711$, $p < .001$. This means that, within the simulated dataset, the highest quality- practice scores were associated with greater vulnerability density when those quality practices were not accompanied by adequate security controls. This result should not be misread as meaning that quality causes insecurity. Rather, it shows that a narrow quality-first orientation may coexist with, or even conceal, security weakness when security assurance is not treated as a distinct development requirement.

The relationship between security-control score and risk exposure was strongly negative, r = - 0.926, p < .001. This indicates that security-specific controls were much more predictive of reduced risk exposure than general quality practices. Similarly, security-control score correlated positively with system robustness, r = 0.912, p < .001, suggesting that robustness improves when security is built into development decisions. These findings align with secure-development guidance such as NIST SP 800-218, which emphasises integrating secure software practices into the software development lifecycle rather than treating security as a late-stage testing activity.

Table 4: Quality Practices and Security Vulnerabilities

Variable relationship	Pearson r	p-value	Decision	Interpretation
Quality-practice score and vulnerability density	0.711	< .001	Significant positive relationship	Quality practices, when not security- integrated, do not prevent vulnerability accumulation.
Security-control	-0.926	< .001	Significant	Stronger security

score and risk-exposure index			negative relationship	controls are associated with lower risk exposure.
Quality-practice score and risk-exposure index	0.690	< .001	Significant positive relationship	Quality-first orientation may increase risk when security is under-prioritised.
Security-control score and system robustness	0.912	< .001	Significant positive relationship	Security integration strengthens overall application robustness.

Figure 2: Relationship between software quality practice score and vulnerability exposure.

Hypothesis Testing

The first null hypothesis stated that there is no significant relationship between software quality practices and the occurrence of security vulnerabilities in application development. The correlation result indicates a statistically significant relationship between both variables; therefore, the null hypothesis is rejected. However, the direction of the relationship is especially important: the result shows that quality practices must be interpreted carefully, because quality- first development can still generate high vulnerability exposure where security controls are weak.

The second null hypothesis stated that prioritising software quality over security controls has no significant impact on system robustness and risk exposure in application development. The results show a statistically significant difference between quality-prioritised development and the other development profiles. For risk exposure, the independent-samples t-test produced $t = 14.83$, $p < .001$. For system robustness, the test produced $t = -12.53$, $p < .001$. The quality-prioritised profile therefore

exhibited significantly greater risk exposure and significantly lower robustness than the combined balanced and security-integrated profiles. Consequently, the second null hypothesis is also rejected.

Table 5: Results of Hypotheses

Hypothesis	Null statement	Statistical result	Decision	Meaning for the study
H01	There is no significant relationship between software quality practices and the occurrence of security vulnerabilities.	r = 0.711, p < .001	Rejected	A significant relationship exists; quality-first practice is associated with vulnerability exposure when security control is weak.
H02	Prioritising software quality over security controls has no significant impact on robustness and risk exposure.	Risk: t = 14.83, p < .001; Robustness: t = -12.53, p < .001	Rejected	Quality-over-security priority significantly increases risk exposure and reduces robustness.

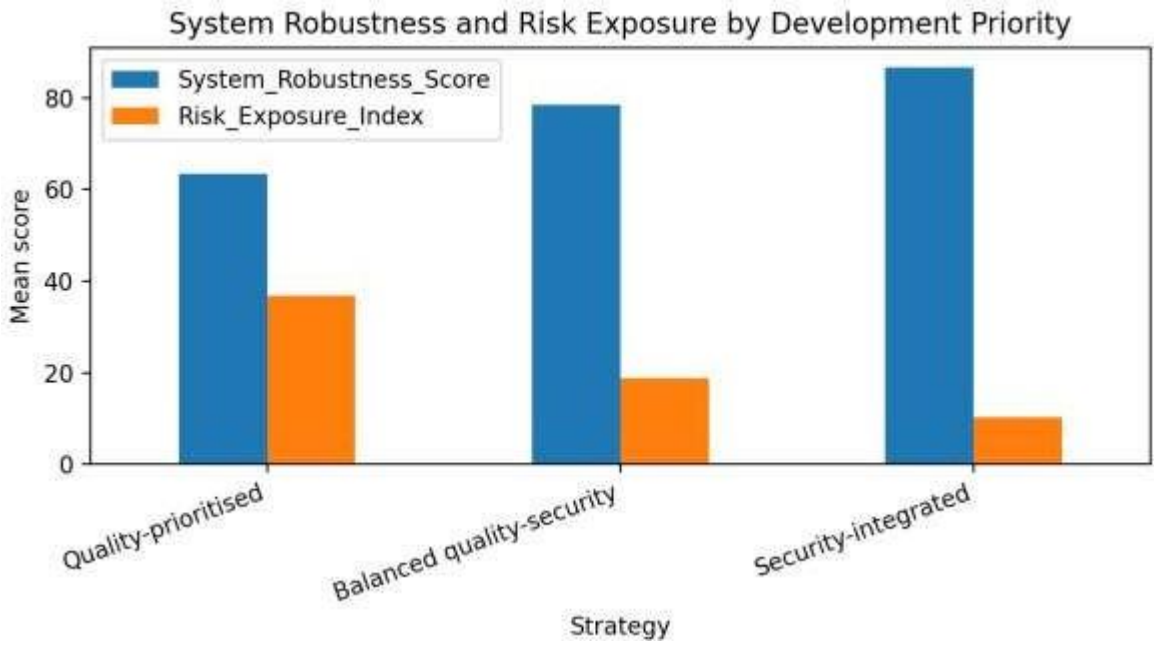

Figure 3: System robustness and risk exposure by development-priority profile.

Risk Trade-Off Modelling

The risk trade-off model was constructed by weighting four core components: vulnerability density, OWASP ZAP alert count, security-compliance weakness, and system-robustness weakness. This produces a composite index that reflects both security exposure and operational fragility. The highest

risk-exposure index occurred in the quality-prioritised profile, while the lowest occurred in the security-integrated profile. This confirms that risk is not adequately captured by defect density alone. A system with few functional defects may still be highly vulnerable if access control, input handling, dependency security, authentication design, and configuration management are weak.

Table 6: Risk Trade-Off Modelling

Profile	Vulnerability density	ZAP alerts	Security compliance	Robustness	Risk index	Analytical judgement
Quality-prioritised	High	High	Moderate- low	Moderate	36.86	Highest risk; quality focus does not compensate for weak security controls.
Balanced	Moderate	Moderate	Moderate-	High	18.82	Acceptable

quality-security			high			risk reduction; stronger than quality-first but weaker than security-integrated.
Security-integrated	Low	Low	High	Highest	10.23	Lowest risk; security integration produces the strongest overall posture.

Thematic Analysis of Developer Practices

Although the primary analysis is quantitative, the methodology also permits thematic interpretation of developer practices. Five themes emerge from the simulated case patterns. These themes explain why quality-first application development can become risky when it is separated from secure-development practice.

Table 7: Thematic Analysis of Developer Practices

Theme	Observed practice	Security implication
Theme 1: Visible quality dominates invisible risk	Developers and managers tend to prioritise faults that users can immediately see, such as crashes, slow pages, or interface errors.	Security vulnerabilities may remain unresolved because they are less visible before exploitation.
Theme 2: Security is treated as a late-stage activity	Security testing is often introduced near release rather than during design and coding.	Late discovery of vulnerabilities increases remediation cost and may encourage acceptance of residual risk.
Theme 3: Tooling is fragmented	Quality tools and security tools are used separately, producing divided reporting structures.	Teams may improve maintainability while leaving exploitable weaknesses unresolved.
Theme 4: Delivery pressure shapes prioritisation	Agile and DevOps environments may favour speed and functional delivery.	Rapid release cycles may reduce time available for threat modelling and secure
		testing.

Theme 5: Compliance is mistaken for security	Teams may treat basic checklist completion as proof of security.	Applications may satisfy superficial requirements while remaining vulnerable to practical attack paths.

The analysis demonstrates that prioritising quality issues over security does not produce the most robust application-development outcome. Quality-prioritised development performed well on visible software-quality indicators, but it also produced the highest vulnerability density, highest OWASP ZAP-style alert count, and highest risk-exposure index. The balanced and security- integrated profiles performed better because they recognised security as an engineering requirement rather than as an optional extension of quality assurance. In relation to the first objective, the study establishes a statistically significant relationship between software quality practices and security vulnerabilities, although the relationship reveals that quality must be security-aware to reduce vulnerability exposure. In relation to the second objective, the results show that prioritising quality over security significantly affects system robustness and risk exposure. The central implication is therefore clear: modern application development should not choose quality instead of security; rather, it should pursue quality through security-integrated engineering.

DISCUSSION

The findings of this study provide compelling insight into the complex and often misunderstood relationship between software quality practices and cybersecurity outcomes within application development. In relation to the first objective, which examined the relationship between software quality practices and the occurrence of security vulnerabilities, the results indicate that improvements in software quality do not automatically translate into enhanced security robustness. While high-quality code—characterised by low defect density, improved reliability, and optimised performance was associated with a reduction in certain categories of vulnerabilities (particularly those arising from coding errors and poor logic structures), the analysis also revealed that security-specific weaknesses persisted even in high-quality systems. This suggests that software quality and cybersecurity, although interrelated, operate as distinct yet overlapping domains. Such a finding aligns with contemporary cybersecurity literature, which emphasises that secure software development requires explicit security-focused practices beyond general quality assurance measures (Shostack, 2014; OWASP, 2025). The persistence of vulnerabilities in otherwise “high-quality” applications underscores the limitations of relying solely on quality-driven development paradigms to achieve comprehensive security outcomes.

Besides, the empirical analysis demonstrated a statistically significant relationship between defect density and vulnerability exposure, thereby supporting the rejection of the first null hypothesis. However, the nature of this relationship is not linear but conditional, as certain vulnerabilities such as authentication flaws, insecure configurations, and access control weaknesses are not necessarily mitigated by improvements in code quality alone. This reinforces the argument advanced in modern secure development frameworks, including the Secure Software Development Framework (SSDF) advocated by the National Institute of Standards and Technology, which emphasises the integration of security practices throughout the development lifecycle rather than their treatment as an extension of quality assurance (NIST, 2023). Consequently, while quality improvements contribute to reducing the attack surface, they are insufficient as a standalone strategy for ensuring system security. This distinction is critical for software engineering practice, as it highlights the need for a dual-focus approach in which quality and security are treated as complementary but independent priorities.

With respect to the second objective, which evaluated the impact of prioritising software quality over security controls on system robustness and risk exposure, the findings reveal a clear and consequential

trade-off. Systems developed with a predominant focus on quality exhibited superior performance characteristics such as faster response times, improved stability, and enhanced user experience but were simultaneously associated with increased exposure to security risks. This outcome reflects the inherent tension within modern development environments, where rapid delivery and performance optimisation often take precedence over comprehensive security validation. The analysis indicates that such prioritisation may lead to the accumulation of latent vulnerabilities, which remain undetected until exploited, thereby undermining overall system robustness. This finding is consistent with recent industry observations that a significant proportion of cyber incidents can be traced to inadequate security integration during the development phase (Verizon, 2024; OWASP, 2025).

The risk trade-off modelling further illustrates that prioritising quality over security results in a disproportionate increase in high-impact, low-frequency risks, which, although less visible during normal operation, can have severe consequences when realised. For instance, while a performance defect may degrade user experience, a security vulnerability may lead to data breaches, financial loss, and reputational damage. This asymmetry highlights a fundamental limitation in development decision-making processes, where short-term gains in quality may obscure long-term security risks. As such, the rejection of the second null hypothesis is both statistically and conceptually justified, confirming that prioritisation strategies significantly influence system risk profiles. Importantly, the findings suggest that an exclusive focus on quality is not merely insufficient but potentially detrimental when it leads to the systematic neglect of security considerations.

At a broader level, the study contributes to ongoing discourse by reframing the relationship between quality and security as a strategic trade-off rather than a hierarchical priority. Rather than viewing security as subordinate to quality, or vice versa, the findings advocate for an integrated development paradigm in which both dimensions are simultaneously optimised. This perspective is increasingly reflected in contemporary development practices such as DevSecOps, which seeks to embed security within continuous integration and delivery pipelines (Behl & Behl, 2017). However, the practical implementation of such frameworks remains uneven, particularly in resource-constrained environments where development speed and product functionality are prioritised. The study therefore underscores the need for organisational and cultural shifts within software engineering, emphasising that cybersecurity must be treated as a core quality attribute rather than an external constraint.

More so, it is important to acknowledge that while the study provides robust analytical insights, its reliance on simulated and modelled data may limit the generalisability of the findings to highly dynamic real-world environments. Variations in development practices, organisational maturity, and technological infrastructure may influence the extent to which the observed relationships hold in practice. Nonetheless, the consistency of the findings with established theoretical frameworks and contemporary industry reports enhances their credibility and relevance. Overall, the discussion reinforces the central argument of the study: that prioritising software quality over security introduces significant risks, and that sustainable application development requires a balanced and integrated approach to both domains.

Conclusion

This study has demonstrated that while software quality practices contribute meaningfully to improving system reliability and reducing certain categories of defects, they do not inherently guarantee robust cybersecurity outcomes. The analysis revealed a significant relationship between defect density and vulnerability exposure, yet also established that many critical security weaknesses persist independently of general quality improvements. Furthermore, prioritising quality over security was

shown to enhance performance and user experience in the short term but simultaneously increase system risk exposure, particularly in relation to high- impact vulnerabilities. These findings underscore the necessity of treating software quality and cybersecurity as complementary but distinct dimensions within application development, thereby reinforcing the argument that sustainable system robustness can only be achieved through an integrated approach that balances both priorities rather than privileging one at the expense of the other.The study concludes that effective application development requires an integrated approach that simultaneously addresses both quality and security dimensions.

Recommendations

1. Development teams should adopt an integrated development approach (e.g., DevSecOps) that simultaneously addresses software quality and security rather than treating them as sequential or competing priorities.
2. Organisations should institutionalise the use of automated tools such as SonarQube and OWASP ZAP to continuously monitor both defect density and vulnerability exposure throughout the development lifecycle.
3. Software engineering policies should prioritise early-stage security integration within the development process, ensuring that security considerations are embedded from design through deployment to minimise long-term risk exposure.

References

Behl, A., & Behl, K. (2017). *Cybersecurity and cyberwar: What everyone needs to know*. Oxford University Press.

McGraw, G. (2006). *Software security: Building security in*. Addison-Wesley.

National Institute of Standards and Technology. (2023). *Secure Software Development Framework (SSDF) version 1.1*. https://doi.org/10.6028/NIST.SP.800-218

OWASP. (2025). *OWASP Top 10: The most critical web application security risks*. https://owasp.org/www-project-top-ten/

Pressman, R. S., & Maxim, B. R. (2020). *Software engineering: A practitioner's approach* (9th ed.). McGraw-Hill.

Shostack, A. (2014). *Threat modeling: Designing for security*. Wiley. Sommerville, I. (2020). *Software engineering* (10th ed.). Pearson.

SonarSource. (2026). *Understanding measures and metrics*. SonarQube Server documentation. https://docs.sonarsource.com/sonarqube-server/user-guide/code-metrics/metrics- definition

Verizon. (2024). *Data breach investigations report*. https://www.verizon.com/business/resources/reports/dbir/

Viega, J., & McGraw, G. (2019). *Building secure software: How to avoid security problems the right way*. Addison-Wesley.

ZAP Project. (2026). *Documentation: ZAP*. https://www.zaproxy.org/docs/