# Effective Integration of Database Security Tools into SDLC Phases: A Structured Framework

**Ahmed Naguib[1,*], Haba K. Aslan[2], Khaled M. Fouad[3,4]**

[1]Faculty of Information Technology and Computer Science, Nile University, Egypt

[2]Faculty of Information Technology and Computer Science, Nile University, Egypt

[3]Faculty of Computers and Artificial Intelligence, Benha University, Egypt

[4]Faculty of Computer Science and Engineering, New Mansoura University, Egypt

Emails: a.naguib2283@nu.edu.eg; haslan@nu.edu.eg; khaled.foad@nmu.edu.eg

**Abstract**

As organizations increasingly rely on digital data, securing database systems has become a critical priority for protecting sensitive information, ensuring system integrity, and meeting regulatory compliance standards. This paper explores a comprehensive framework for database security, focusing on developing, assessing, and testing effective security tools. We begin by outlining the essential steps in creating robust security tools, including defining specific requirements based on database types and access needs and implementing real-time monitoring systems for immediate threat detection. The paper also emphasizes the importance of regular vulnerability assessments and advanced security analytics to identify and address potential risks proactively. Insights from a recent survey conducted among database administrators revealed that key areas of concern include access control, real-time monitoring, and vulnerability assessments. Furthermore, we highlight the significance of integrating security practices throughout the Software Development Life Cycle (SDLC). Additionally, best practices for evaluating and testing database security, including penetration testing to uncover vulnerabilities and stress testing to assess performance under load, are discussed. By synthesizing these strategies and survey feedback, this paper provides a comprehensive approach to enhancing database security, ensuring data protection, and maintaining system resilience against evolving cyber threats

**Keywords:** Database Security; Encryption; Access Control; Vulnerability Assessments; Real; Time Monitoring; Penetration Testing; Data Confidentiality; Data Integrity; Compliance Standards; Risk Management

## 1. Introduction

In today's technology-driven world, databases are the backbone of virtually every organization, underpinning various applications and services critical to business operations. As the volume and sensitivity of the data they manage grows, so does the imperative to secure these valuable assets against an evolving landscape of cyber threats. Adequate database security is no longer a mere technical requirement but a fundamental aspect of an organization's risk management strategy and operational integrity [1].

The importance of robust database security is underscored by the increasing complexity of the System Development Life Cycle (SDLC) models used to design, develop, and maintain these systems. Traditional SDLC models like Waterfall provided a structured, systematic approach to software development but often struggled to accommodate modern applications' dynamic and iterative nature. This has led to the adopting of more flexible frameworks, such as Agile and Iterative models, which offer iterative development and continuous feedback, aligning better with the fast-paced and evolving demands of today's technology environment [2][3].

Securing databases effectively involves more than implementing security measures; it requires a comprehensive understanding of database security goals and challenges. The primary objectives are to ensure data confidentiality, integrity, and availability (CIA). Confidentiality restricts data access to authorized users; integrity ensures that data remains accurate and unaltered, and availability guarantees that data is accessible when needed. These goals



DOI: https://doi.org/10.54216/JCIM.160114
Received: November 03, 2024 Revised: January 17, 2025 Accepted: February 12, 2025

are crucial for protecting sensitive information from unauthorized access, maintaining compliance with regulatory standards, and preventing data breaches that could have severe financial and reputational repercussions [4][5].

The consequences of inadequate database security can be devastating. Data theft, damage to business reputation, revenue loss, and increased operational costs are all potential outcomes of security breaches. Organizations must adopt a multi-faceted approach to mitigate these risks, including advanced encryption techniques, stringent access controls, and regular security audits. Furthermore, securing data during transfer—especially when migrating to cloud environments—requires careful implementation of encryption and network security protocols [6][5][7][8].

In addition to traditional security measures, modern database security strategies must address the unique challenges relational databases pose and emerging cybersecurity threats. While Relational Database Management Systems (RDBMS) offer structured data management, they also present specific vulnerabilities that must be addressed through client-side protections and comprehensive security strategies [9][10].

The System Development Life Cycle (SDLC) plays a critical role in the security of databases. The SDLC encompasses various phases, each with its unique security considerations. These phases include planning, analysis, design, implementation, testing, deployment, and maintenance. Each phase offers opportunities to embed security measures that can help protect the database from potential threats. Identifying security requirements and possible risks is crucial during the planning and analysis phases. The design phase should incorporate security features into the architecture, while the implementation phase must ensure secure coding practices. The testing phase is essential for identifying and addressing vulnerabilities, and the deployment phase must ensure that security configurations are correctly applied. Finally, maintenance involves continuously monitoring and updating security measures to address emerging threats [2][11].

The impact of poor database security extends beyond immediate financial losses. It can lead to long-term reputational damage, legal liabilities, and loss of customer trust. Businesses must recognize that adequate database security is an ongoing process that requires continuous evaluation and improvement. As cyber threats evolve, so must the strategies and technologies used to defend against them. Organizations must stay informed about the latest developments in cybersecurity and invest in training and resources to maintain a robust security posture [12][13][5].

Another significant aspect of database security is the integration of security practices with regulatory compliance. Organizations are subject to various regulations, such as the General Data Protection Regulation (GDPR), the Health Insurance Portability and Accountability Act (HIPAA), and the Payment Card Industry Data Security Standard (PCI DSS). These regulations mandate stringent data protection measures, and non-compliance can result in hefty fines and legal consequences. By aligning database security practices with regulatory requirements, organizations protect their data and ensure compliance with legal standards, avoiding potential legal and financial repercussions [4] [14].

Moreover, the rise of cloud computing presents both opportunities and challenges for database security. Cloud environments offer scalability, flexibility, and cost savings but also introduce new security concerns. Data stored in the cloud is susceptible to threats different from those stored in on-premises databases. Organizations must implement robust security measures, such as encryption, access controls, and continuous monitoring, to protect cloud-based databases. Additionally, they must carefully evaluate the security practices of cloud service providers to ensure that their data remains secure in the cloud [5][10] [14].

In support of this, we surveyed database administrators and IT professionals, revealing that their top concerns were integrating access control mechanisms, securing databases during data transfer, and ensuring regulatory compliance. These findings align with the primary goals of database security: ensuring data confidentiality, integrity, and availability (CIA). The survey also highlighted a growing need for real-time monitoring tools and frequent vulnerability assessments to identify and address potential risks proactively. These concerns are compounded by the increasing complexity of System Development Life Cycle (SDLC) models used to build and maintain secure database systems. Traditional models, like Waterfall, provided a structured approach to development. Still, modern iterative models, such as Agile and Rapid Application Development (RAD), offer the flexibility to address these evolving security requirements effectively. This paper integrates the findings from the survey with an in-depth discussion of these models, security strategies, and tools designed to protect sensitive data from external and internal threats.

The role of emerging technologies in enhancing database security cannot be overlooked. Innovations such as artificial intelligence (AI) and machine learning (ML) are being leveraged to improve security measures. AI and ML can analyse vast data to identify patterns and anomalies indicating potential security threats. These technologies can enhance threat detection and response, enabling organizations to address security issues before they escalate proactively. Furthermore, blockchain technology is being explored for its potential to provide secure and immutable records of database transactions, further strengthening database security [10][5].



DOI: https://doi.org/10.54216/JCIM.160114
Received: November 03, 2024 Revised: January 17, 2025 Accepted: February 12, 2025

This paper provides a holistic approach to database security, embedding security measures throughout the Software Development Life Cycle (SDLC). The study introduces encryption, access control, and real-time monitoring techniques to fortify database security. One of the paper's key contributions is integrating a survey that examines the use of database security tools in various SDLC phases, offering insights from professionals across industries. Moreover, this paper emphasizes two critical aspects: the human factor in achieving security objectives and integrating project management methods with database security. The research presents a practical framework for improving collaboration, embedding security early in development processes, and aligning security goals with business and technical requirements by focusing on these dimensions.

The rest of the paper is organized as follows: Section 2 contains a comprehensive review of System Development Life Cycle (SDLC) models, highlighting their impact on security practices. Then, the literature review is detailed in Section 3. Section 4 depicts a survey on database security practices across SDLC phases; next, the proposed model for database security in the SDLC is illustrated in Section 5. Section 6 discusses the limitations of the study. Finally, the paper concludes with future work in Section 7.

## 2. Background

The System Development Life Cycle (SDLC) is critical in guiding software development processes, evolving to meet modern business needs. Different SDLC models, such as Waterfall, Agile, and Rapid Application Development (RAD), offer distinct advantages and disadvantages depending on project requirements. Security integration throughout the SDLC is essential, particularly for database systems, to ensure data protection, compliance, and resilience against evolving cyber threats. This section examines various SDLC models, explores database security goals, and outlines the challenges and strategies for securing relational databases (RDBMS) across the software development lifecycle.

### A. System Development Life Cycle (SDLC) Models: Advantages and Disadvantages

The System Development Life Cycle (SDLC) has evolved significantly, driven by advancements in software development concepts and the increasing demand for customer-centric applications and solutions. Historically, software development adhered to a more rigid structure, but modern SDLC frameworks are designed to be adaptable, meeting the diverse needs of organizations with varying backgrounds, structures, and goals [2]. This adaptability ensures that regardless of an organization's unique policies and procedures, there are shared objectives at each stage of system development, allowing for minimal variations in how projects are described, organized, and managed across different contexts. This flexibility makes today's SDLC frameworks suitable for various businesses, products, and services.

Among the various models within the SDLC framework, several are particularly noteworthy due to their widespread use and unique characteristics. The Waterfall model, for instance, follows a linear sequence of stages, each dependent on the completion of the previous one [3]. This model is characterized by its structured approach, making it easy to understand and manage, especially for projects with well-defined requirements and a clear scope. However, its rigidity can present significant challenges when requirements evolve during the project lifecycle.

In contrast, the iterative and agile models incorporate flexibility and continuous feedback, allowing for ongoing adjustments throughout the development process. These models are particularly beneficial for projects with evolving requirements or those requiring rapid delivery of functional components [13]. The iterative model focuses on gradual improvement through repeated cycles, enabling incremental enhancements and refinements.

The agile model emphasizes rapid delivery and continuous improvement, dividing the product into smaller builds, each introducing new features with regular feedback and adjustments.

One of the key goals in integrating security practices into agile development methodologies is identifying security needs in the early stages of the SDLC. This process allows security technologies such as data encryption and strict security policies to be incorporated into the design phase while ensuring secure coding standards are applied during development. Comprehensive testing, including stress and penetration tests, is conducted in the testing phase to ensure the system can withstand threats. Continuous maintenance and security updates are vital to keeping data safe and preventing modern threats [15].

The Rapid Application Development (RAD) model prioritizes swift development and iterative prototyping, focusing on the rapid delivery of software products rather than extensive planning and documentation. This approach expedites the software development process by concentrating on creating small, functional components in a highly collaborative environment. RAD encourages continuous user involvement and feedback, enabling developers to quickly adapt to changing requirements and deliver a product that closely aligns with user expectations [10].

Understanding the advantages and disadvantages of these various SDLC models is crucial for selecting the most appropriate method for a specific project. Key comparison areas typically include process structure, flexibility and



DOI: https://doi.org/10.54216/JCIM.160114
Received: November 03, 2024 Revised: January 17, 2025 Accepted: February 12, 2025

adaptability, customer involvement, risk management, testing and quality assurance, project size and complexity, documentation and traceability, time and cost, team collaboration and communication, and security considerations. By comparing SDLC models across these dimensions, decision-makers can choose a framework that best fits the project's specific needs and context, ensuring successful software development outcomes [4].

### B. Database Security Goals: Ensuring Confidentiality, Integrity, and Availability

Database security is vital for protecting an organization's critical data. The main goals are ensuring data confidentiality, integrity, and availability (CIA). Confidentiality restricts access to authorized users only, integrity ensures data remains accurate and reliable, and availability guarantees data access when needed [5]. Table 1 highlights the main objectives of database security.

**Table 1:** Key Aspects of Database Security

| Category | Details |
|---|---|
| Security Goals | Protect confidentiality, integrity, and availability (CIA). |
| Sensitive Data Protection | Use encryption and access controls to prevent unauthorized access. |
| Compliance | Adhere to regulations like GDPR, HIPAA, and PCI DSS to avoid penalties. |
| Preventing Breaches | Implement vulnerability scans, encryption, and access controls. |

### C. The Impact of Poor Database Security: Threats and Consequences

In today's digital age, database security is crucial for protecting sensitive data. As businesses store increasing volumes of information, the risks of data breaches and cyberattacks grow. Effective database security ensures data confidentiality, integrity, and availability, protecting it from human errors, excessive privileges, insider threats, and malware attacks.

- What is Database Security, and Why is it Important?

Database security involves measures to protect databases from threats. It includes securing data, database management systems, applications, servers, and network infrastructure. The aim is to prevent unauthorized access and maintain data confidentiality, integrity, and availability [16].

The table below (Table 2) highlights the potential consequences of poor security practices.

**Table 2:** Consequences of Poor Database Security

| Category | Impact |
|---|---|
| Data Theft | This leads to identity theft and financial loss. |
| Reputation Damage | Loss of customer trust and sales decline. |
| Revenue Loss | Business disruption and potential regulatory fines. |
| Increased Costs | Legal fees, recovery expenses, and security upgrades. |
| Regulatory Penalties | Fines and compensation for affected individuals. |

This layout provides a concise, clear overview of both the goals and potential negatives.



DOI: https://doi.org/10.54216/JCIM.160114
Received: November 03, 2024 Revised: January 17, 2025 Accepted: February 12, 2025

### D. Enhancing RDBMS Security: Exclusive Focus on Relational Databases

Relational Database Management Systems (RDBMS) such as SQL are the backbone for managing and securing extensive data repositories in the modern digital landscape. Ensuring end-to-end security in these systems is paramount. While most RDBMS platforms provide robust server-side and in-transit data protection, significant vulnerabilities often arise on the client side. This paper addresses these gaps by proposing a methodology that extends security measures to encompass client-side protection, thereby maintaining the confidentiality, integrity, and availability of sensitive data throughout its lifecycle [9].

Effective database security necessitates a comprehensive strategy to mitigate various risks, including human errors, excessive database privileges, insider threats, malware, and physical server damage. By integrating server-side protections, secure data transmission, and robust client-side security measures, organizations can safeguard critical data assets such as customer records, financial information, and intellectual property against these threats.

Relational databases, with their structured schema and emphasis on data integrity, are often the preferred choice for storing sensitive information. This paper highlights the importance of RDBMS security, examining inherent strengths, and addressing client-side vulnerabilities. Additionally, it compares relational databases with non-relational databases, which offer flexibility and scalability but may lack some intrinsic security features [10].

As organizations navigate the complexities of data management, understanding the distinct advantages of relational and non-relational databases is crucial for selecting the appropriate system based on specific security needs. This discussion paves the way for future research and development in database security, advocating for a holistic approach that ensure robust protection across all levels of data management.

### E. Security Strategies during Data Transfer

In today's digital landscape, safeguarding the secure transfer of data is crucial, especially when migrating sensitive customer information to cloud environments. This guide presents essential strategies to uphold data confidentiality, integrity, and availability throughout the transfer process [5].

Firstly, encrypting data at rest is fundamental. Advanced Encryption Standard (AES) encryption and secure key management ensure that sensitive data stored in databases remains inaccessible to unauthorized access, even if physical storage media is compromised.

Secondly, configuring network services with SSL/TLS protocols is imperative. These protocols encrypt data in transit, protecting it from interception during transmission between on-premises systems and cloud platforms. This robust encryption layer is pivotal in securing sensitive or proprietary information during exchange.

Moreover, using tools as if Database Data Pump with encryption enabled facilitates secure data transfers. This approach ensures data integrity and confidentiality while moving large volumes of information across different environments or networks [8].

Establishing Virtual Private Network (VPN) connections further enhances security by creating encrypted tunnels for data traffic across public or less secure networks. VPNs protect sensitive data from unauthorized access and maintain confidentiality during transit.

Data masking techniques in non-production environments also obscure sensitive information, preventing inadvertent exposure while complying with stringent privacy regulations [17].

Lastly, implementing regular audits and monitoring using tools like Oracle Audit Vault and Database Firewall is essential. These tools provide continuous oversight of data access and usage, enabling swift detection and response to potential threats or policy violations.

Organizations can fortify their data transfer processes by adopting these comprehensive security measures, adhering to regulatory standards, and cultivating trust with stakeholders in an evolving digital landscape [14].

Securing the transfer of sensitive data between different environments involves implementing security techniques at every stage of the SDLC. In the requirements phase, security protocols such as AES encryption are identified. In the design phase, security policies are incorporated to protect data in transit through networks using protocols like SSL/TLS. During development, secure coding practices are applied to ensure that no security vulnerabilities exist. The testing phase focuses on verifying security through comprehensive tests to protect data during transmission. Finally, maintenance requires continuous monitoring to ensure data integrity and protection from new threats.

### F. Database Security Roles and Strategies across the Software Development Life Cycle (SDLC)

In today's digital age, databases hold vast amounts of sensitive information, making them prime targets for cyberattacks. Protecting this data is crucial for ensuring data integrity, confidentiality, and compliance with



DOI: https://doi.org/10.54216/JCIM.160114
Received: November 03, 2024 Revised: January 17, 2025 Accepted: February 12, 2025

regulatory requirements throughout its entire lifecycle. The Database Security Life Cycle (DSLC) consists of phases such as risk assessment, security design, implementation, monitoring, and maintenance, all aimed at safeguarding data from unauthorized access or theft. Similarly, the Software Development Life Cycle (SDLC) plays a critical role in ensuring a project's success and security, embedding security considerations from the initial requirements gathering to maintenance. Properly defined roles and responsibilities in both DSLC and SDLC ensure that security measures are integrated at every stage, reducing risks and ensuring compliance with security standards [18].

The SDLC phases—Requirements, Design, Development, Testing, Deployment, and Maintenance—contribute to building a secure and resilient system. Security needs such as data protection and compliance standards are identified early in the requirements phase. The Design phase integrates these security requirements into the system architecture, including encryption and access control measures. During the development phase, secure coding practices are applied to prevent vulnerabilities, and security tests are conducted to verify compliance.

Testing involves rigorous evaluations like penetration testing to uncover potential security flaws before deployment. The deployment phase ensures that security configurations are appropriately set as the system transitions to production. Finally, maintenance involves continuous monitoring, applying patches, and updating security policies to address emerging threats and maintain long-term data protection.

This comprehensive approach across both DSLC and SDLC ensures that database security is prioritized at every stage, reducing risks and safeguarding sensitive information throughout the software development process [18].

### G. Current Challenges in Database Security and Software Development

In today's technologically driven world, databases are the lifeblood of organizations, housing vast amounts of critical and sensitive information. This pivotal role makes them prime targets for cyberattacks, highlighting the necessity for robust security measures to protect data integrity, confidentiality, and compliance with regulatory standards. As digital technologies become more deeply integrated into business operations, the complexity of database security challenges intensifies, compelling organizations to adopt comprehensive and adaptive security strategies [12].

The landscape of cyber threats is rapidly evolving, with attackers continuously devising new methods to exploit vulnerabilities. The repercussions of inadequate security measures can be severe, ranging from significant financial losses and reputational damage to legal liabilities. Enterprises must proactively implement encryption, access controls, auditing, and continuous monitoring to safeguard their databases [7].

Several common security issues pose significant database risks, including excessive user privileges, SQL injections, malware, weak audit trails, backup exposure, and weak authentication. Each of these issues necessitates targeted solutions to mitigate risks effectively. Limiting user privileges, securing backup storage, implementing strong authentication mechanisms, and maintaining comprehensive audit trails are some of the best practices to address these challenges [12].

Moreover, database vulnerabilities and misconfigurations, unmanaged sensitive data, and a lack of security expertise further exacerbate security risks. Regularly updating and patching systems, implementing strict data management policies, and investing in security training are crucial to fortifying database security. Data breaches have heightened for years recently, with millions of records compromised by hackers, insider threats, and human errors, making robust security practices indispensable [16].

Addressing these challenges requires a multifaceted approach integrating technical, administrative, and physical controls. The CIA triad—confidentiality, integrity, and availability—remains a foundational principle for protecting sensitive data. By continuously improving and adapting security strategies, organizations can mitigate risks, ensure compliance, and maintain their database systems' long-term resilience and reliability against evolving cyber threats [12].

Organizations must also foster a culture of security awareness among employees to prevent accidental data leaks and insider threats. This includes regular training on security best practices, recognizing potential threats, and understanding the importance of following security protocols. Creating a security-conscious culture within an organization is crucial for minimizing risks and ensuring that security measures are consistently upheld [16].

In conclusion, securing databases in today's digital landscape requires addressing various challenges and implementing comprehensive security measures. By integrating encryption, access controls, auditing, monitoring, and fostering a culture of security awareness, organizations can protect their sensitive data, comply with regulatory requirements, and maintain stakeholder trust. The dynamic nature of security threats underscores the need for continuous improvement and adaptation in security strategies, ensuring database systems' long-term resilience and reliability. Establishing and maintaining robust data security measures is essential for shielding sensitive



DOI: https://doi.org/10.54216/JCIM.160114
Received: November 03, 2024 Revised: January 17, 2025 Accepted: February 12, 2025

information, preserving trust, and ensuring regulatory adherence in an increasingly complex digital environment [12].

### H. Comprehensive Database Security: Techniques and Tools

Key techniques such as encryption encoding data at rest and in transit using advanced algorithms are crucial for maintaining confidentiality and compliance with regulations like GDPR and HIPAA. Similarly, access control mechanisms such as role-based access control (RBAC) and biometrics restrict data access to authorized users, enhancing security and ensuring auditability. Techniques like inference policy and data confidentiality safeguard information from unauthorized access by preventing disclosure through data correlation and inference.

Critical tools include Data Safe, which provides security assessments, user risk scoring, activity auditing, and Database Vault, which restricts access to sensitive data based on policy controls. Data Redaction dynamically masks sensitive data, while Data Masking anonymizes data in non-production environments, maintaining security during testing. Tools such as Password Check enforce strong password policies, and Code Block ensures that only authorized code is executed in production. For securing data at rest, Transparent Data Encryption (TDE) encrypts databases without requiring application changes. Additionally, Database Audit tools log and monitor database activities, providing insights into user behaviour and enabling the detection of suspicious activity.

Securing relational databases (RDBMS) requires a comprehensive approach that includes protecting data at every phase of the SDLC. This involves identifying security requirements early, such as during the analysis phase, where potential risks are assessed. In the design phase, security is integrated into the architecture, enforcing encryption and access control policies. During development, secure coding techniques are adopted to reduce security vulnerabilities. Penetration tests and threat assessments are conducted in the testing phase to ensure the system is safe before deployment. The maintenance phase ensures ongoing monitoring and security updates for long-term data protection."

Organizations can implement a robust and comprehensive database security strategy by integrating encryption, access control, masking, and auditing tools, ensuring their data is protected from internal and external threats, maintaining regulatory compliance, and mitigating risks throughout the entire database lifecycle.

## 3. Literature Review

Previous research indicates that the security and success of software development projects depend on a comprehensive approach that integrates effective project management, human factors, secure coding practices, data protection techniques, and continuous monitoring throughout the Software Development Life Cycle (SDLC). Studies have emphasized that safeguarding sensitive data through encryption and masking, identifying security risks, and ensuring compliance are critical in mitigating vulnerabilities and delivering reliable software products [18]. This literature review examines various methodologies and tools that contribute to enhanced security, including the roles of project management and human awareness, the significance of encryption and data masking, and the integration of security tools such as Audit Vault, Data Safe, and Transparent Data Encryption (TDE) to ensure data confidentiality and integrity. As highlighted in prior studies, each component plays a crucial role in reinforcing security and maintaining project success across different stages of the SDLC.

Several recent contributions have provided effective methodologies for secure software development and testing, emphasizing the integration of security practices throughout the SDLC. One such methodology is a model-based secure development and testing approach incorporating DevSecOps principles, presented in Computers & Security. According to this study, the approach facilitates early threat identification and countermeasure selection through static assessments and targeted security tests. A case study on a microservice-based application demonstrates the methodology's effectiveness in detecting vulnerabilities and implementing mitigation strategies through comprehensive test plans [19].

Furthermore, previous studies have highlighted that secure coding practices are critical in preventing vulnerabilities like code injection attacks during the SDLC. Researchers suggest that adopting secure coding standards enforced by tools like Code Block ensures that unauthorized code execution is stopped. As recommended in the literature, continuous code reviews and static analysis tools further aid in identifying and addressing potential security issues early in the development process, thereby improving the overall integrity of the application [15].

Another key factor in software security identified in the literature is enforcing strong password policies. As noted in previous studies, tools such as Password Check enforce password complexity standards, ensuring adherence to best practices for password management. Research suggests that regularly updating password policies, educating users on the importance of strong passwords, and implementing multi-factor authentication are essential measures to prevent unauthorized access and security breaches [13].

Studies have also emphasized integrating security measures and monitoring tools throughout the SDLC to maintain applications' security, compliance, and reliability. As indicated in prior research, monitoring data activities and



DOI: https://doi.org/10.54216/JCIM.160114
Received: November 03, 2024 Revised: January 17, 2025 Accepted: February 12, 2025

implementing relevant tools, such as Audit Vault and Database Firewall, are essential for continuous oversight [11]. These tools log and audit activities in real time, ensuring that organizations can detect and respond to security incidents as they occur. According to some studies, regular audits and reviews of security policies can identify vulnerabilities and areas for improvement, thus enhancing the overall security posture [11].

Audit Vault has been identified in the literature as a key player in continuous database monitoring. Research indicates that its comprehensive auditing capabilities, including detailed audit trails, enable organizations to monitor all database activities [20]. As previous studies have shown, this helps comply with regulatory requirements, detect suspicious activities, and mitigate potential security risks by implementing appropriate countermeasures [20].

In addition to security monitoring, accurate application testing plays a significant role in ensuring system reliability and security, as highlighted in the literature. Studies demonstrate that real application testing replicates production workloads to identify potential issues before deployment, thus preventing production failures [15]. This proactive approach, as discussed in prior research, improves the overall quality of the software by ensuring that applications perform as expected in live environments, reducing operational risks [15].

Numerous studies emphasize that protecting sensitive data throughout the SDLC is critical to secure software development. Various techniques, such as data masking, subsetting, and redaction, are widely used to safeguard information during development and testing [14,17]. According to the literature, these methods ensure organizations can work with realistic datasets while maintaining data privacy and regulatory compliance, reducing the risk of data breaches and unauthorized access [14,17].

Data masking is particularly crucial during the development phase. Previous research indicates that it allows organizations to protect sensitive information by replacing it with fictional but structurally similar data [17]. Tools like Data Masking and Subsetting offer comprehensive solutions to anonymize data in non-production environments, ensuring that organizations meet data protection regulations while maintaining a secure development environment, as highlighted in studies [17].

During the testing and design phases of the SDLC, data redaction assumes a significant role in concealing specific data elements. The literature suggests that organizations can effectively prevent unauthorized access to critical data by dynamically masking sensitive information while allowing developers to work with realistic datasets [17]. When used collectively, these techniques help organizations protect sensitive information, ensure compliance with data protection regulations, and mitigate the risk of data breaches throughout the SDLC, as previous studies have shown [17].

In the evolving cybersecurity landscape, encryption has emerged as one of the most critical strategies for ensuring data protection across various stages of the SDLC. Researchers emphasize that whether data is in transit or at rest, encryption technologies provide a robust defense against unauthorized access and cyber-attacks, offering organizations a reliable means of safeguarding sensitive information [8].

Data encryption during transmission is vital for protecting information from interception and tampering. As the literature notes, protocols like SSL/TLS establish encrypted connections between senders and receivers, ensuring that data transmitted over the Internet remains confidential and secure [5]. This method is significant for safeguarding sensitive data, such as financial transactions and personal information, from cyber-attacks. Studies suggest that organizations must prioritize encryption in transit to ensure the security and privacy of their communications [5].

Encryption is not just a solution but also an optimal one for database security, ensuring the confidentiality and integrity of data both in transit and at rest. Transparent Data Encryption (TDE) is a key technology for protecting data at rest by encrypting database files at the table or column level, as identified in previous research [8]. Even if unauthorized users gain access to storage media, TDE ensures that data remains protected, thus safeguarding intellectual property and meeting regulatory requirements for data security [8]. As an additional layer of security, TDE maintains data confidentiality and prevents unauthorized access, providing a comprehensive and robust approach to data protection that instils confidence in organizations, as discussed in the literature [8].

Project management and human factors are pivotal in software development and IT project success. The literature indicates that effective project management methodologies and a focus on human awareness and training are essential for maintaining security and ensuring project completion that meets technical and security requirements.

According to previous studies, project management is integral to the success of IT projects, particularly within the SDLC [2,15]. It ensures that projects are delivered on time, within budget, and at the required quality. Audit Vault monitors and logs activities throughout the project, ensuring compliance with security standards across all phases, as identified in the research. Adopting methodologies like Agile and Scrum enhances flexibility and responsiveness, allowing for continuous improvement and incremental deliveries that meet evolving customer



DOI: https://doi.org/10.54216/JCIM.160114
Received: November 03, 2024 Revised: January 17, 2025 Accepted: February 12, 2025

needs [23]. Additionally, Database Vault enforces strict access policies, assigning permissions based on roles to enhance data security further, as noted in the literature [2,15].

Similarly, the human factor in data security is paramount, as emphasized in previous research. Studies indicate that human errors often lead to significant security breaches, making it essential for organizations to invest in continuous employee training and awareness programs [1,7]. Educating staff about cybersecurity risks can help reduce security incidents, while strict policies and procedures help control access to sensitive data. Tools like Database Vault enforce strict access policies, assigning permissions based on roles to enhance data security further, as highlighted in the literature. This combined approach to managing human and technical factors strengthens organizations' overall security posture [1,7].

Identifying and addressing security risks throughout the SDLC is vital to ensuring secure, reliable software creation. Modern database and software security approaches emphasize continuous risk management, security testing, and monitoring as essential components of a robust security posture, as discussed in various studies [21].

Data Safe, a comprehensive security tool, is identified in the literature as an effective solution for identifying and mitigating database security risks [14]. Offering features such as data discovery, assessment, and monitoring, Data Safe enables organizations to proactively address vulnerabilities throughout the data lifecycle. According to research, organizations can protect sensitive data by providing better visibility into their security posture, enhancing their overall data security framework [14].

As previous studies have shown, integrating risk assessments at each phase of the SDLC is critical. This proactive approach enables organizations to identify and resolve potential issues early in development, ensuring that the final product meets user expectations and complies with security requirements. Researchers have highlighted key concerns in global software development, such as lack of threat modeling, insufficient output validation, and inadequate certification, using the fuzzy analytical hierarchy process (FAHP) to prioritize risks [22]. Addressing these critical risks is key to fortifying the security of global software projects, as emphasized in the literature.

Furthermore, an integrated security-testing framework within the Secure SDLC (SSDLC) can significantly enhance enterprise security. Studies demonstrate that by adopting automated security test cases and enforcing specific security guidelines, organizations can ensure quality and stable services in their software development processes [19]. Such frameworks have shown the potential to improve software products' overall security and reliability when implemented effectively, as indicated in prior research [19].

In conclusion, integrating security practices into the SDLC has been the subject of considerable research. Many studies have highlighted the importance of incorporating security measures to mitigate risks and enhance the robustness of software systems. However, a critical analysis reveals that most of this research focuses on specific tools or phases of the SDLC, such as testing or risk assessment, without providing a comprehensive framework that details the security techniques appropriate for each phase. This literature review identifies the gaps in these studies. It explains how the present research addresses these gaps by proposing a structured framework with suitable security techniques and tools for each SDLC phase.

**Gap Analysis of Previous Research**

Several significant gaps have been identified in reviewing the existing literature on integrating security practices within the Software Development Life Cycle (SDLC). These gaps highlight areas where previous research has been limited or lacks comprehensive guidance, underscoring the need for a more holistic approach to database security in the SDLC. The following sections detail these gaps under specific headings.

A. Limited Focus on Specific Phases (Testing)

One prominent gap in the literature is the concentration on the testing phase to the exclusion of other SDLC phases. For instance, Mothanna et al. [24] extensively focus on security testing frameworks for sustainable smart cities. While their research provides valuable insights into how testing can identify and mitigate security vulnerabilities, it remains confined to this single phase. The lack of guidance on security techniques for other SDLC phases means potential vulnerabilities could be introduced earlier in the development process and remain undetected until testing, which may be too late for cost-effective remediation.

B. Emphasis on Risk Management Without Specific Implementation Techniques

Another gap is the emphasis on risk identification and assessment without offering specific security techniques for implementation. Fisher and White [7] discuss effective risk management strategies within each SDLC phase, highlighting the importance of recognizing potential threats. However, their research stops short of providing actionable techniques that developers and security professionals can apply to mitigate identified risks in each phase. This leaves a gap between understanding risks and knowing how to address them practically within the development lifecycle.



DOI: https://doi.org/10.54216/JCIM.160114
Received: November 03, 2024 Revised: January 17, 2025 Accepted: February 12, 2025

C. Concentration on Specific Security Tools Without Integration Across SDLC

Research has also focused on individual security tools without integrating them into a comprehensive SDLC framework. For example, Mousa et al. [17] conducted an in-depth survey on data masking techniques, and Smith and Garcia [5] explored advanced data encryption methods for secure cloud storage. While these studies contribute valuable knowledge about specific tools, they do not offer guidance on how they can be applied across different SDLC phases. This siloed approach prevents practitioners from understanding how to integrate various security tools effectively throughout development.

D. Security Practices in Agile Methodologies Limited to Agile Models

Some studies have addressed integrating security practices within specific development methodologies but lack generalizability to other models. For instance, Anderson and White [15] examine how security can be embedded within agile development processes. Their research is valuable for teams using agile methodologies but does not provide detailed security techniques applicable to each phase of other SDLC models like Waterfall. This limitation means organizations using different development models may not benefit from their findings.

E. Overlooking Security Techniques in Early SDLC Phases

A critical gap identified is the general trend of overlooking security techniques during the early phases of the SDLC, such as requirements gathering and design. This neglect can lead to potential vulnerabilities being introduced at the very beginning of the development process. Without proper guidance on security practices during these initial phases, developers may make foundational decisions that compromise security, which are often costly and complex to rectify later in the SDLC. The absence of security focus in the early stages undermines the overall security posture of the final product.

While previous studies have contributed significantly to understanding various security aspects in software development, they often focus on specific tools or phases, leaving gaps in comprehensive security integration. Our research addresses these gaps by providing a structured framework that integrates security techniques and appropriate tools into each phase of the SDLC. This approach fills the void identified in the literature and offers practical solutions for enhancing database security throughout the software development process.

The motivation behind this research stems from the critical need to enhance security integration within the Software Development Life Cycle (SDLC). The gaps identified in the literature review revealed that most existing research focuses on specific tools or phases, such as testing or risk assessment, without providing a comprehensive framework that outlines security techniques for each phase. This fragmented approach can lead to potential vulnerabilities, as security is not consistently addressed throughout development.

We aim to develop a practical and holistic framework that maps out security techniques and appropriate tools for each SDLC phase. By doing so, we address the following motivations:

A. Improving Security Practices Across the SDLC

By integrating security techniques into every phase, we aim to prevent vulnerabilities from being introduced early in the development process and ensure that security considerations are an integral part of the project's lifecycle.

B. Providing Practical Guidance for Practitioners

Offering a detailed framework with specific tools and techniques helps practitioners effectively implement security measures, overcoming the lack of actionable guidance in previous research.

C. Enhancing Overall Software Quality

Integrating security throughout the SDLC not only improves the security posture of the software but also contributes to higher quality and reliability, as potential issues are identified and addressed early.

This research is driven by the potential to affect how organizations approach security in software development significantly. By providing a comprehensive and practical framework, we aim to promote a proactive security culture and enhance the overall effectiveness of security measures within the SDLC.

## 4. Survey on Database Security Practices across SDLC Phases

To software development methodologies, collaboration between academic research and industrial practice in database security is often limited. This gap is evident in many industries, where research innovations may not always translate into practical implementation, and industry needs may not always be reflected in academic studies. Several surveys and studies have aimed to bridge this gap by offering insights into the tools, techniques, and challenges faced by professionals in this field. These efforts help foster better collaboration between academia and



DOI: https://doi.org/10.54216/JCIM.160114
Received: November 03, 2024 Revised: January 17, 2025 Accepted: February 12, 2025

industry, aligning research with real-world needs and ensuring practitioners access the latest advancements in database security.

In finance, healthcare, and government sectors, database security plays a crucial role in maintaining trust and system reliability. The global IT industry faces ongoing challenges in delivering secure software solutions while balancing performance and cost. The rise of data breaches and cyberattacks underscores the importance of implementing robust database security practices.

This survey aims to provide a comprehensive understanding of the current state of database security practices. It seeks to identify the tools, techniques, and methodologies professionals use and highlight the challenges they encounter when implementing these practices. By understanding these aspects, we aim to provide valuable insights for practitioners and researchers, ultimately fostering stronger collaboration between academia and industry.

This survey explored using tools such as encryption, masking, and auditing while addressing common obstacles to fully implementing security measures.

### A. Survey goal, design, and execution

This section presents the survey's goals, design, and execution.

- Goal and Research Questions

The primary goal of this survey is to evaluate the practices, challenges, and tools related to database security within the Software Development Life Cycle (SDLC). Insights from professionals in various industries help to identify how security measures are implemented and the obstacles hindering full adoption. The following research questions (RQs) were formulated:

**RQ1:** What are the profiles of the participants, including their academic background, current roles, and industry sectors?

**RQ2:** What are the main challenges preventing teams from fully implementing security measures, such as encryption, masking, or auditing?

**RQ3:** During which phases of the SDLC do participants believe security tools are most effectively used, and how does this impact project success?

**RQ4:** What are the most commonly used tools for database security, and how effective are they in ensuring the confidentiality, integrity, and availability of data?

**RQ5:** How does familiarity with software development methodologies (e.g., Waterfall, Agile) influence team security practices?

- Sampling Method

The survey employed a convenience sampling approach, selecting participants based on accessibility and willingness to participate. While this introduces some bias, it is widely accepted for exploratory studies. The survey focused on individual professionals working in database security and software development.

- Survey Questions

The survey consisted of 27 questions designed to capture data related to participants' backgrounds, the tools they use, and the challenges they face. Examples include:

1. Which tools do you consider the most effective for each phase of the Software Engineering process?
2. What are the main challenges preventing full implementation of security measures?
3. Which phase is recommended for assessing and evaluating potential security risks that should be addressed in the subsequent stages of the SDLC?

- Survey Execution

The survey was distributed via email and social media channels like LinkedIn and WhatsApp to maximize outreach. Participants were encouraged to share the study within their professional networks. The survey remained open for 3 Days, during which data from 50 participants was collected and analyzed. Partial responses were included, as they provided valuable insights.

### B. Survey results and findings

This section presents and analyzes the survey results based on the responses from the participants. The key findings for each research question are as follows:

- What is your highest academic degree?



DOI: https://doi.org/10.54216/JCIM.160114
Received: November 03, 2024 Revised: January 17, 2025 Accepted: February 12, 2025

This question is aimed at determining the educational qualifications of participants. Understanding the highest academic degree participants hold provides insight into their academic backgrounds, which may influence their approach to software development and security practices. For instance, individuals with advanced degrees such as a master's or PhD may have specialized knowledge in cybersecurity. In contrast, those with bachelor's degrees might focus on more practical, industry-related security aspects.

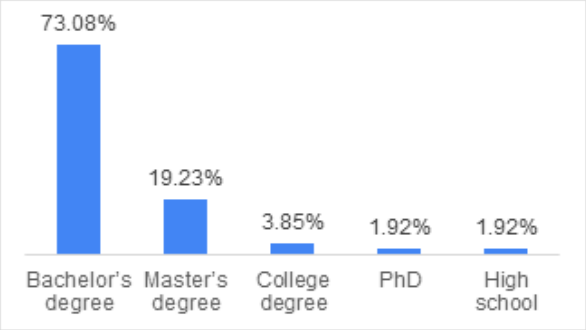


**Figure 1.** Highest academic degree of participants.

Figure 1 illustrates the distribution of academic degrees among participants. The majority (73%) hold a bachelor's degree, indicating that most respondents have received a foundational IT and software engineering education. Additionally, 19% have obtained a master's degree; while a smaller portion, (3.85%) possess a college degree. Only one participant holds a PhD, and another has completed High School as their highest level of education. This highlights that the survey respondents generally have a strong academic foundation, with many pursuing higher educations in technical fields.

- What types of industries have the products you have worked on been for?

This question seeks to understand the industries participants have worked in, such as IT, Telecommunications, Healthcare, Financial Services, and others. Different sectors have distinct security standards and requirements, influencing how participants approach security challenges. For instance, sectors like healthcare and financial services often require stringent compliance with data privacy regulations, which could shape the participants' focus on security protocols and technologies.

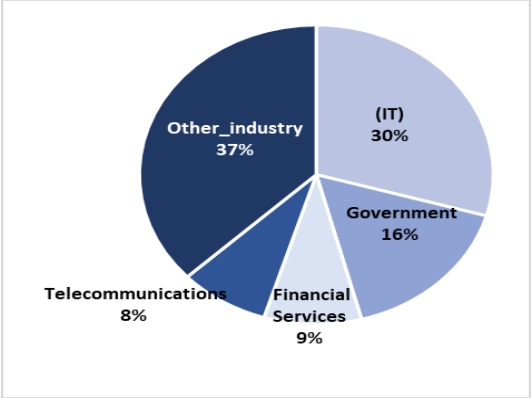


**Figure 2.** Distribution of industries in which participants have worked.

The pie chart in Figure 2 shows that 30% of participants have worked in the IT industry, commonly involving infrastructure development and software solutions that require robust security measures. Meanwhile, 16% of respondents have experience in government roles, where regulatory compliance and secure data management are crucial. Additionally, 9% have worked in Financial Services, a sector known for its high data protection and encryption standards due to legal and consumer demands. Telecommunications accounted for 8%, highlighting the importance of secure communication systems. Finally, 37% of participants come from other industries, including healthcare, education, and retail sectors, each presenting unique security challenges.

- How many years of work experience do you have in the IT and software development industries?



DOI: https://doi.org/10.54216/JCIM.160114
Received: November 03, 2024 Revised: January 17, 2025 Accepted: February 12, 2025

This question aims to gauge the participants' professional experience in the IT and software development industries. The years of experience often correlate with a participant's expertise in managing complex projects, troubleshooting, and addressing security challenges. By understanding the levels of experience, we can better analyze responses from newer and more seasoned professionals, allowing us to interpret the findings based on varying levels of exposure to industry practices.

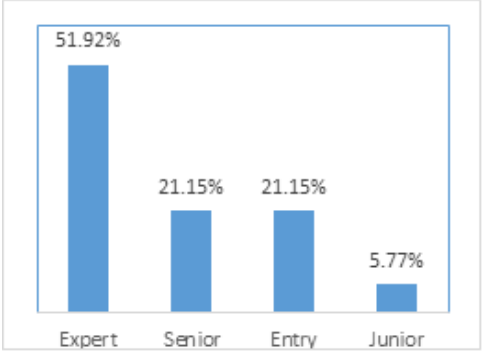


**Figure 3.** Distribution of participants by career level.

The chart in Figure 3 reveals that over 52% of participants with more than 11 years of professional experience fall into the Expert category. This significant representation of experienced individuals suggests that seasoned professionals who are likely to have considerable insight into advanced security practices and challenges heavily inform the survey responses. Additionally, 22% of participants are classified as Senior, followed by 22% at the Entry level. Only 3% of respondents identified as junior professionals, indicating a smaller proportion of individuals with less than 2 years of experience in the field.

- What is the size of the companies you have worked for in terms of the number of employees?

This question aims to understand the organizational scale of the companies the participants have worked for. The size of a company often influences its approach to software development and security practices. Larger organizations generally have more resources to implement sophisticated security measures and adopt best practices. At the same time, smaller companies may face constraints in budget or staff, potentially limiting their ability to employ comprehensive security systems. This information provides insight into participants' work environments and helps contextualize their perspectives on security challenges and solutions.

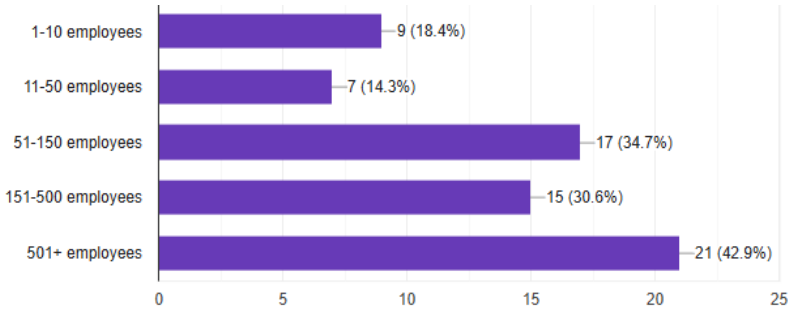


**Figure 4.** Company size based on the number of employees.

The data in Figure 4 shows that 43.8% of participants have worked for companies with 501+ employees, reflecting experience in large organizations where resources for security and development tend to be more substantial. Additionally, 35.4% of respondents have worked in companies with 51-150 employees, while 31.3% have been part of organizations with 151-500 employees, indicating significant representation in mid-sized companies. On the smaller end, 16.7% have worked in organizations with 1-10 employees, and 14.6% have experience in companies with 11-50 employees, highlighting the diversity in company sizes among participants.

- Rate the security team's attention in each SDLC phase in your previous projects.



DOI: https://doi.org/10.54216/JCIM.160114
Received: November 03, 2024 Revised: January 17, 2025 Accepted: February 12, 2025

This question rates the security team's involvement across different phases of the Software Development Life Cycle (SDLC), including requirements, design, development, testing, deployment, and maintenance. The aim is to assess how well integrated security practices are throughout the software development process. Higher security involvement in the early stages, such as requirements and design, reflects a proactive approach to mitigating security risks from the outset. Conversely, low participation in the early stages may indicate that security is addressed only later in the process, potentially missing critical security considerations.


**Figure 5.** Security involvement in different SDLC phases.

In the requirements phase, as depicted in Figure 5, over 66% of participants reported security attention, with most rated it as moderate. This could be a concern, as a strong security focus from the outset is critical to prevent vulnerabilities from being built into the system. A moderate focus here could introduce risks to the overall security posture of the product.

During the testing, phase (see Figure 5), more than 77% of participants indicated high attention from the security team, highlighting a strong emphasis on ensuring system robustness before deployment. This is a positive sign, but focusing on security only on testing might suggest a reactive approach, leading to missed opportunities for early security integration.

The design phase, illustrated in Figure 5, saw 62% of participants reporting moderate security involvement. While this shows some consideration of security during planning, the absence of a stronger focus could indicate a risk. Designing systems with security in mind early can help prevent vulnerabilities and reduce the need for costly rework later.

In the development phase, as shown in Figure 5, more than 75% of participants noted security team involvement, which is encouraging, as it suggests that security is being addressed during coding. However, continuous feedback between development and testing is essential to catch vulnerabilities early.

Finally, over 82% of respondents reported strong security attention during the maintenance phase (refer to Figure 5), a favorable outcome given the importance of ongoing support in securing systems after deployment. However, the high focus on maintenance might imply that security issues are being discovered later in the process, which can be costly and potentially dangerous.

- Assessment and Risk Implications:

The survey results show both positive and concerning trends. While it is encouraging that security is a focus during the testing, development, and maintenance phases, the moderate involvement during the requirements and design phases presents a potential risk. Security should ideally be integrated throughout the SDLC, starting from the early stages, to ensure a comprehensive and proactive approach. Relying too heavily on testing and maintenance to identify security issues can lead to costly fixes and leave room for vulnerabilities during the early stages of development.



DOI: https://doi.org/10.54216/JCIM.160114
Received: November 03, 2024 Revised: January 17, 2025 Accepted: February 12, 2025

Thus, the results suggest that while some organizations take security seriously, there is stillroom for improvement in early-stage security integration. The moderate attention in the requirements and design phases is a red flag that could indicate underlying vulnerabilities that may surface later in the lifecycle.

- Based on your experience, at what stage of the SDLC does sensitive data typically first appear?

This question investigates when sensitive data is typically introduced into the Software Development Life Cycle (SDLC). The timing of when sensitive data first appears is crucial because it determines when security measures must be implemented. Suppose sensitive data is introduced early in the process (such as during development or design). In that case, there is a greater need to have strong security protocols in place from the outset to protect against data breaches. On the other hand, if sensitive data only appears later (such as during deployment), there may be a smaller risk window. , the need for strong data protection measures remains critical.

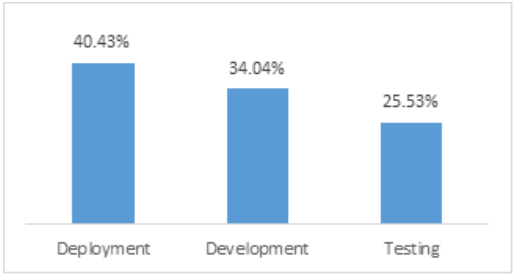


**Figure 6.** Stages of the SDLC where sensitive data first appears.

- Deployment phase: Over 40% of participants reported that sensitive data first appears during deployment.
- Development phase: 33% indicated that sensitive data first appears during development.
- Testing phase: 25% noted that sensitive data is introduced during testing.

The survey results, depicted in Figure 6, show that many participants report sensitive data appearing first in the deployment phase. This finding could signal a potential risk because security measures might not be robustly integrated earlier in the SDLC. It could leave room for vulnerabilities if sensitive data is introduced late in the cycle without proper security preparations in the development and testing stages.

On the other hand, if data is introduced earlier, such as during the development phase, there is more opportunity to enforce security protocols throughout the entire lifecycle. The fact that many participants also indicated that sensitive data appears during the development phase shows that some organizations may be taking proactive steps. Still, the dominant focus on the deployment stage suggests that more work is needed to secure earlier phases of the SDLC.

In conclusion, while some organizations may address security when sensitive data first appears, focusing on the deployment stage raises concerns about potential exposure earlier in the development process. This emphasizes the need for continuous security vigilance and the implementation of early security measures throughout the SDLC.

- Based on your experience and knowledge, which phase is recommended for assessing and evaluating potential security risks that should be addressed in the subsequent stages of the SDLC?

This question focuses on determining which phase of the SDLC is most suitable for identifying and evaluating potential security risks. The earlier security risks are assessed, the easier it becomes to address these issues before they become deeply embedded in the system architecture. Identifying risks during the requirements or design phases can result in more cost-effective security solutions, while identifying risks later in the development or deployment stages may lead to expensive, time-consuming fixes and possible security vulnerabilities.

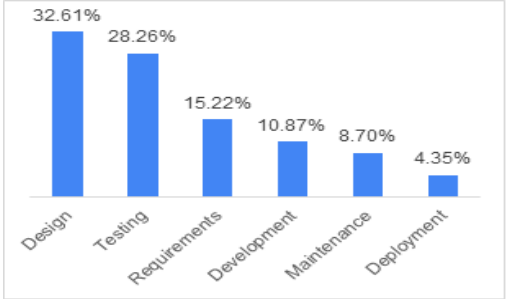


**Figure 7.** Recommended SDLC phase for assessing security risks.



DOI: https://doi.org/10.54216/JCIM.160114
Received: November 03, 2024 Revised: January 17, 2025 Accepted: February 12, 2025

- Design phase: Over 32% of participants recommend this phase as the most suitable for identifying and addressing security risks.
- Testing phase: 28% of respondents selected this phase as a preferred phase for security assessments.
- Requirement phase: Chosen by 15%, indicating early-stage involvement in risk evaluation.
- Other stages (Development, Maintenance, and Deployment): Less frequently chosen, with the deployment phase being the least recommended.

The results in Figure 7 highlight a focus on addressing security risks during the design phase, a positive indicator of proactive security practices. However, the preference for the testing phase also shows that many participants consider security assessments during later stages, potentially indicating some reliance on reactive security measures. Ideally, more attention should be given to the requirements phase to ensure a comprehensive security strategy is developed early in the process.Top of FormBottom of Form

- Based on your experience, what is the most commonly used tool in database security?

This question seeks to identify the database security tools most frequently used by participants, such as Encryption (TDE), Database Vault, Masking, or Auditing. Understanding which tools are most commonly used can provide insight into the industry with the best practices for securing databases and how effectively these tools address common security threats.

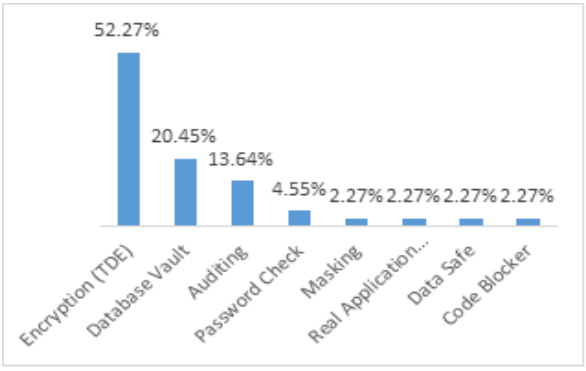


**Figure 8.** Most commonly used tools in database security.

- Encryption (TDE): Chosen by 52% of participants as the most used tool for database security.
- Database Vault: Selected by 20% of participants, highlighting its role in controlling database access.
- Auditing: Identified by 13%, showing its importance in monitoring database activities.
- Other Tools: Tools like Password Check, Masking, and Real Application Testing (RAT) were less frequently used.

The dominance of TDE (Transparent Data Encryption) as the most commonly used database security tool is a positive indicator, reflecting the industry's focus on protecting data at rest. However, as shown in Figure 8, the relatively lower adoption of tools like Auditing and Database Vault might suggest gaps in monitoring database activity or controlling privileged access, which could be a potential risk for some organizations.

- Which recommended tool is used to monitor and log database activities to detect and respond to suspicious or unauthorized actions?

This question focuses on tools like Database Vault, Real Application Testing (RAT), or Auditing, which monitor and log database activities. These tools are critical for detecting and responding to unauthorized access or suspicious actions, providing an audit trail that can be used for forensic analysis in case of a security breach.

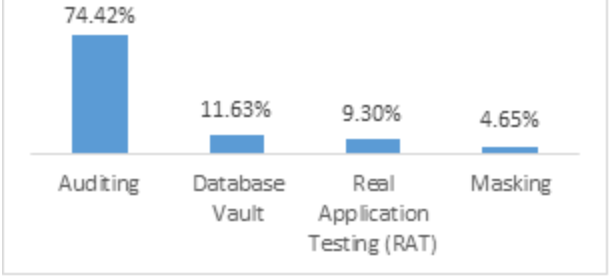


**Figure 9.** Tools used to monitor and log database activities.



DOI: https://doi.org/10.54216/JCIM.160114
Received: November 03, 2024 Revised: January 17, 2025 Accepted: February 12, 2025

- A significant majority selected auditing, with 74% of participants recognizing it as the most effective tool for monitoring and logging database activities.
- Database Vault and RAT were less frequently chosen, with 11% and 9% of respondents, respectively, indicating their more specialized roles in database security.
- Masking received minimal attention, with only 4% of participants selecting it, reflecting its use for securing sensitive data rather than monitoring activities.

The preference for Auditing as the primary tool for monitoring database activities underscores its essential role in database security, providing real-time insights into unauthorized actions and helping organizations respond swiftly to potential breaches. As illustrated in Figure 9, while tools like Database Vault and RAT have important functionalities, Auditing remains the cornerstone for comprehensive activity logging and incident response within databases.

- What kind(s) of software development methodologies do you use in your team?

This question identifies which software development methodologies are currently used by participants' teams. Knowing whether teams use Waterfall, Agile, or Hybrid methodologies helps analyze how security and development practices are integrated, particularly in modern development frameworks.

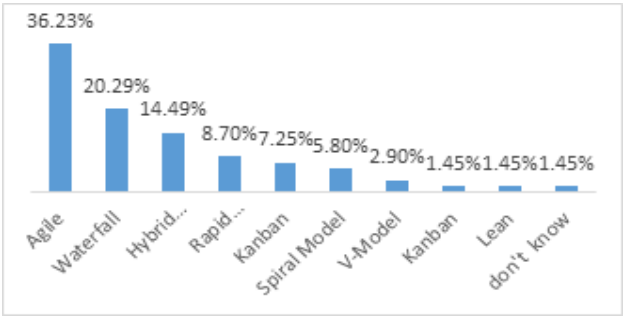


**Figure 10.** Software development methodologies used by participants' teams.

- 36.23% of participants favor the agile methodology, highlighting a trend toward iterative and flexible development.
- 20.29% of participants still use Waterfall, which suggests a more traditional, phased approach to development.
- 14.49% of participants use Hybrid Methodologies, combining aspects of different methodologies to adapt to specific project needs.

The results depicted in Figure 10 show the dominance of Agile as the preferred methodology, used by over a third of participants, demonstrating a substantial shift toward flexibility in development processes. However, the continued use of Waterfall by 20.29% of teams indicates that specific projects or organizations may still require a more structured, phase-driven approach. Hybrid methodologies offer a balance, allowing teams to adapt and integrate different practices to suit unique project requirements. The blend of methodologies suggests that security practices will need to be equally adaptable, ensuring they are robust across varied workflows.

- What are the main challenges or obstacles prevent you and your team from fully implementing security measures?

This question addresses the common challenges teams face in implementing security measures, such as high costs, complexity, lack of expertise, or resistance to change. Understanding these obstacles is critical for identifying where additional support or resources are needed to achieve comprehensive security practices.

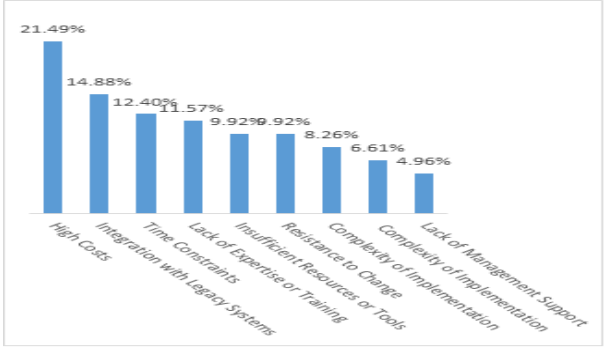


**Figure 11.** Common challenges preventing the full implementation of security measures.



DOI: https://doi.org/10.54216/JCIM.160114
Received: November 03, 2024 Revised: January 17, 2025 Accepted: February 12, 2025

- High costs were reported as the most significant obstacle, with 21.49% of participants indicating it as a challenge.
- Integration with legacy systems is another key issue, affecting 14.88% of respondents.
- Time constraints and lack of expertise or training were reported by 12.40% and 11.57% of participants, respectively.
- Other challenges include insufficient resources, resistance to change, and implementation complexity.

The challenges shown in Figure 11 emphasize that high costs and integration with legacy systems are the most common obstacles preventing teams from fully implementing security measures. This suggests that financial investment and system compatibility are critical factors influencing the adoption of robust security practices. Addressing these issues will require increased funding, targeted training, and careful planning to manage legacy system integrations without compromising security.

- What test technique did the team use to generate test cases in your current or most recent software project?

This question focuses on the methods participants' teams use to generate test cases, such as Manual Test Case Design, Automated Test Case Design, or Model-Based Testing. The goal is to understand which approaches are most used and how these methods contribute to ensuring software quality and security. Automated testing, for example, can enhance efficiency and coverage, while manual testing may provide more nuanced insights in specific scenarios.

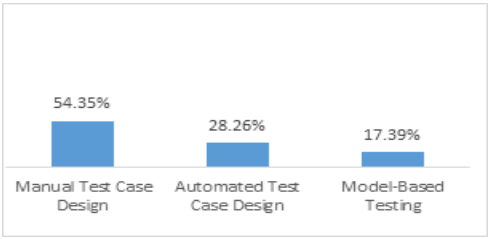


**Figure 12.** Test techniques are used to generate test cases.

- 54% of participants use Manual Test Case Design.
- 28% of participants rely on Automated Test Case Design.
- 17% of participants reported using Model-Based Testing.

As shown in Figure 12, manual test case design predominance highlights that many teams still rely on this traditional method despite advancements in automated testing tools. Although manual testing can provide detailed insights, transitioning to automated and model-based testing could enhance the efficiency and coverage of software testing practices, especially in environments requiring rapid iterations.

- From your experience, during which phase of the Software Engineering process do you believe each tool is most effectively used?

This question aims to gather insights from participants on when specific security tools are most effectively implemented during the SDLC. By identifying the phases (such as Testing, Deployment, and Maintenance) in which tools like Data Safe, Database Vault, Code Block, Real Application Testing (RAT), Transparent Data Encryption (TDE), and Password Check provide the most value, participants can share best practices and their experiences regarding the impact of these tools on security and performance. Understanding which tools are considered most effective for each phase helps identify industry standards for ensuring security throughout the entire software development process.

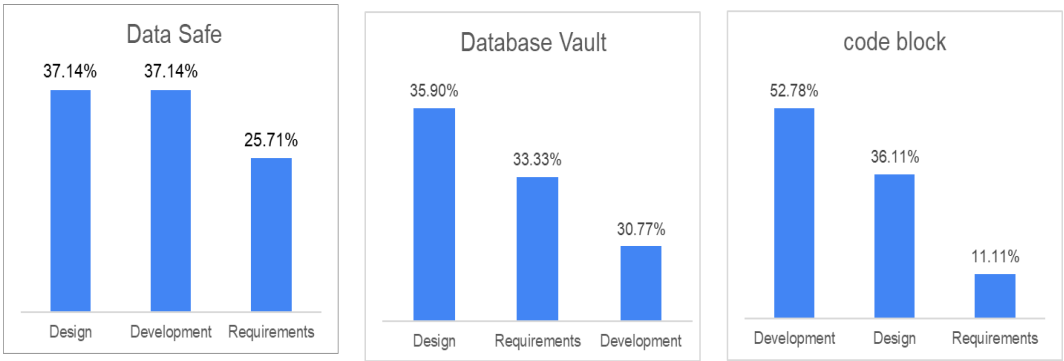


**Figure 13.** Effectiveness of Security Tools across Different Phases of the Software Engineering Process.



DOI: https://doi.org/10.54216/JCIM.160114
Received: November 03, 2024 Revised: January 17, 2025 Accepted: February 12, 2025

As illustrated in Figure 13, Data Safe is considered equally effective during the Design and Development phases, with 28% of participants selecting each phase.

In contrast, Database Vault appears to be most effective in the Design phase, with 35% of respondents identifying it as the optimal stage for its use.

Lastly, Code Block strongly prefers the Development phase, with 52% of participants finding it most beneficial during that stage.

These insights help establish industry standards for ensuring security across the various phases of the software development process.

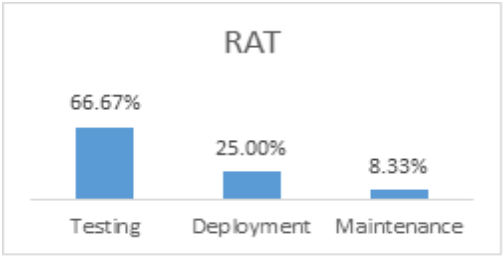


**Figure 14.** Effectiveness of RAT in Software Phases

In the testing phase, a large majority of participants (66%) indicate that RAT is most effective, which aligns with the design of RAT to test applications under real-world conditions before deployment (see Figure 14). In the deployment phase, RAT can still validate the application post-release, but its usage decreases. During the maintenance phase, RAT proves to be less valuable, reflected by the low response count in this area.

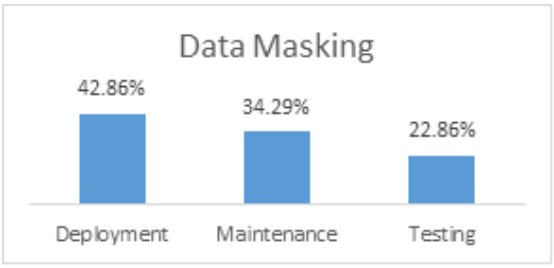


**Figure 15.** Effectiveness of Data Masking in Software Phases

Regarding data masking, deployment is often where sensitive data is exposed, making it critical; over 41% of participants find it most effective at this stage (refer to Figure 16). The maintenance phase is also vital for data masking, mainly when various teams for monitoring or updates access data. Additionally, many participants find data masking important during testing to protect sensitive information in test environments.

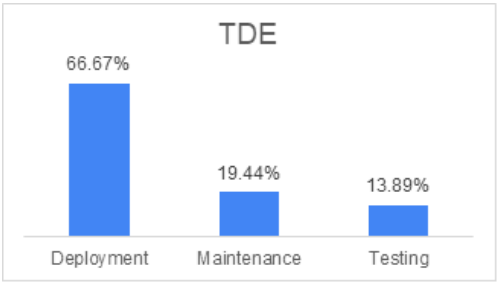


**Figure 16.** Effectiveness of TDE in Software Phases

For TDE, most participants (66%) identify the deployment phase as the most effective for its use, highlighting the importance of encryption when sensitive data is launched in a live environment (see Figure 17). While some participants acknowledge the maintenance phase for TDE, this lower response suggests that encryption is typically



DOI: https://doi.org/10.54216/JCIM.160114
Received: November 03, 2024 Revised: January 17, 2025 Accepted: February 12, 2025

implemented earlier and monitored during maintenance. Usage of TDE during testing is less common, indicating that many teams may rely on other methods that do not necessitate encryption at this stage.

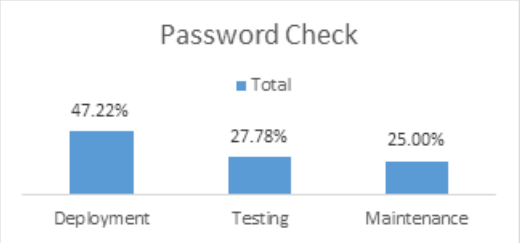


**Figure 17.** Effectiveness of Password Check-in Software Phases

Regarding password checks, most participants agree that they are crucial during deployment, as the application is being prepared for real-world use, making strong password policies essential. Participants also emphasize the importance of ensuring password mechanisms function correctly in testing, which accounts for a reasonable number of positive responses. However, it is understandable that fewer participants focus on password checks during maintenance, even though this practice remains necessary for ongoing security (refer to Figure 17).

The findings highlight a clear trend in the application of security tools during the early and middle phases of the SDLC. Tools like Data Safe and Database Vault are widely regarded as highly effective during the Design phase, emphasizing the importance of securing systems from the outset. Code Block is primarily utilized during the Development phase, reflecting its critical role in preventing the introduction of harmful code. These results suggest that applying security tools at key stages of the SDLC helps mitigate security risks before they become deeply embedded, underscoring the importance of proactive security measures.

While the results align with the industry's best practices, there are some outliers, particularly with tools like TDE and RAT, where a small portion of participants may not fully adhere to standard security protocols in earlier phases such as testing. This suggests that while the majority follow established guidelines, there may be room for improvement in the timing and consistency of security tool applications across the SDLC.

- How would you rate the effectiveness of the suggested model, which involves incorporating security early in the SDLC, if it were applied to software applications?

This question asks participants to evaluate the effectiveness of a security model that incorporates security measures early in the SDLC. Participants rate the model from "Highly Ineffective" to "Highly Effective," providing valuable feedback on whether early-stage security integration is beneficial in preventing security vulnerabilities later in the process.

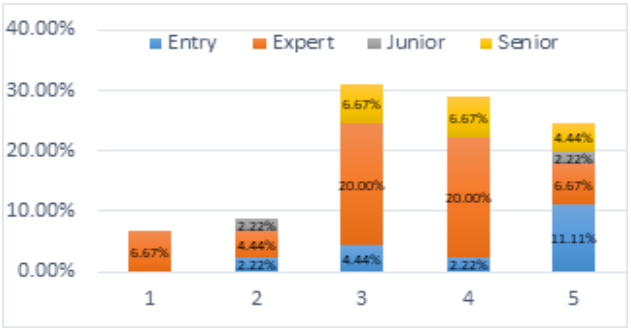


**Figure 18.** Evaluate the effectiveness of the proposed model

- Over 82% of participants rated the model as applicable, assigning it a score of 3 or higher.
- Ratings are distributed across various experience levels, with experts and senior-level professionals predominantly supporting the model.

As shown in Figure 18, the overwhelmingly positive response to incorporating security early in the SDLC suggests that participants recognize the value of early-stage security integration. This proactive approach is beneficial for preventing vulnerabilities before they escalate, reducing the cost and effort of security fixes later in the process. The feedback from experienced professionals further emphasizes that early security measures are a best practice in the software development lifecycle.



DOI: https://doi.org/10.54216/JCIM.160114
Received: November 03, 2024 Revised: January 17, 2025 Accepted: February 12, 2025

### C. Discussion of the survey results

- Summary of Findings

The findings revealed that most participants are involved in the maintenance and deployment phases of the SDLC, with a strong focus on securing databases during these stages. Tools like Transparent Data Encryption (TDE) and Database Vault were widely used, but challenges such as cost and legacy system integration persist.

- Lessons Learned

The survey highlighted that while organizations prioritize security, early security integration in the SDLC remains an area that requires more focus. Many respondents indicated room for improvement in collaboration between academia and industry to address emerging security challenges.

- Threats to Validity

Potential threats include sampling bias due to convenience sampling and incomplete responses. While partial data was used, some findings may have been skewed. Steps taken to mitigate these threats included cross-referencing survey data with existing research.

## 5. Suggested framework for database security in SDLC

In today's increasingly complex digital landscape, ensuring robust database security throughout the Software Development Life Cycle (SDLC) is crucial. Data breaches and cyberattacks are becoming more sophisticated and frequent, posing significant risks to organizations across all sectors. As applications handle increasing amounts of sensitive data, the need for a comprehensive and proactive approach to database security has never been more critical.

This section introduces a comprehensive model to enhance database security across all SDLC phases. By integrating targeted security tools and practices into each phase—from requirements gathering to maintenance—the proposed framework aims to create a resilient and compliant system. This approach addresses traditional security concerns and aligns with modern regulatory requirements, such as the General Data Protection Regulation (GDPR) and the Health Insurance Portability and Accountability Act (HIPAA), ensuring that security is integral to the development process.

The findings inform the framework of our survey on database security practices, which highlighted several critical gaps and challenges in current industry practices. Notably, the survey revealed that security measures are often not integrated early enough in the SDLC, with only a minority of professionals considering security during the requirements and design phases. Challenges such as high implementation costs, integration difficulties with legacy systems, and a lack of expertise hinder the full adoption of essential security practices.

The proposed framework seeks to bridge the gap between best practices and real-world applications by directly addressing these issues. It emphasizes the importance of early and continuous integration of security measures, tailored tool selection for each SDLC phase, and alignment with industry standards and regulatory compliance.

This section outlines the roles and responsibilities of each SDLC phase, presents key security tools suited to address specific challenges, and demonstrates how they collectively contribute to a secure, efficient, and compliant database environment. By embedding security considerations into every phase of the SDLC, organizations can proactively mitigate risks, reduce the likelihood of costly security breaches, and enhance overall system integrity.

**Impact Analysis of Security Tools:**

The statistical analysis presented here is based on data collected from the database environment before and after implementing the suggested security model. The study focuses on the impact of various security tools on overall security posture, system performance, regulatory compliance, and user satisfaction across the SDLC phases. Results indicate significant improvements in identifying and mitigating vulnerabilities, maintaining compliance with data protection regulations, and increasing stakeholder confidence. This evidence underscores the effectiveness of a holistic approach to database security and validates the integration of targeted security measures throughout the development process.

### A. Requirements Phase during SDLC

Data Safe was extensively used in the Requirements phase to assess security risks and ensure compliance with regulatory standards from the outset. This tool helped classify sensitive data, identify potential security risks, and set up appropriate security controls. By integrating Data Safe early in the project lifecycle, the organization experienced a significant reduction in unidentified risks, as potential vulnerabilities were identified and mitigated early on. This proactive approach ensured that the project started with a strong security foundation, significantly enhancing compliance adherence and setting a precedent for robust security practices throughout the SDLC [14].



DOI: https://doi.org/10.54216/JCIM.160114
Received: November 03, 2024 Revised: January 17, 2025 Accepted: February 12, 2025

Additionally, using Data Safe enabled the organization to continuously monitor and evaluate the security posture of their data, providing real-time insights into potential threats and ensuring that all identified risks were addressed promptly. This comprehensive approach to risk management ensured that the project met all necessary regulatory requirements from the beginning, preventing costly delays or rework later in the development process [20].

Furthermore, incorporating Data Safe during the requirements phase facilitated effective communication among stakeholders. The development team, security analysts, and business stakeholders aligned their objectives by clearly defining security and data protection requirements early on. This alignment minimized misunderstandings and ensured everyone understood the security expectations, which is crucial for successfully integrating security measures throughout the SDLC.

Early adoption of Data Safe also allowed the organization to prioritize security features based on risk levels. The team could focus on protecting the most critical assets by classifying data according to sensitivity and potential impact. This risk-based approach to security ensured that resources were allocated efficiently, addressing the most significant threats first and providing the greatest return on investment in risk reduction.

Moreover, integrating security considerations at the requirements stage helped establish a security-centric culture within the organization. It emphasized the importance of security from the beginning, encouraging all team members to consider the security implications of their work. This cultural shift can lead to practices that are more vigilant, such as secure coding and regular security reviews, further strengthening the organization's security posture.

The proactive use of Data Safe also streamlined compliance reporting. Since regulatory requirements were considered from the outset, the necessary documentation and evidence for compliance audits were readily available. This readiness eased the audit process and reduced the risk of non-compliance penalties and the associated reputational damage.

In addition, the early identification of security requirements enabled the organization to anticipate and plan for potential integration challenges with existing systems. By understanding the security needs upfront, the team could design solutions compatible with legacy systems, mitigating one of the common obstacles highlighted in the survey—integration with legacy infrastructure.

By embedding Data Safe into the requirements phase, the organization addressed the survey findings that indicated a lack of early security integration. This approach demonstrated a commitment to shifting security "left" in the development process, which reduces costs and improves security outcomes. It set a strong foundation for the subsequent phases, where security considerations were already integral to the project's DNA.

Using Data Safe during the requirements phase enhanced the project's security, compliance, improved project management, and resource utilization. It underscored the importance of early security integration in developing secure, reliable, and compliant software systems.

### B. Design Phase during SDLC

During the Design phase, Database Vault was implemented to integrate security directly into the system architecture. This tool provided robust security controls by enforcing least privilege access policies and segregation of duties. By embedding these controls into the database, the organization observed a notable reduction in unauthorized access attempts. Database Vault's ability to define and enforce strict access policies ensured that sensitive data was protected from the outset, thereby reducing the risk of insider threats and enhancing overall data security [20].

In addition to implementing Database Vault, the design phase included detailed threat modelling and architectural risk analysis to identify and address potential security weaknesses early. By incorporating these practices, the organization could anticipate and mitigate risks before they could be exploited, ensuring the database architecture was secure and resilient. This proactive approach not only improved the overall security posture of the project but also ensured that all stakeholders were aware of and prepared for potential security challenges [1, 24].

Furthermore, integrating security measures during the design phase facilitated better collaboration between development and security teams. Regular meetings and workshops were conducted to align security requirements with system functionalities, leading to a more cohesive and secure design. This alignment helped minimize conflicts between functionality and security, which is often a challenge in later stages of development.

The early adoption of Database Vault also addressed compliance requirements effectively. By establishing stringent access controls and detailed auditing capabilities from the design phase, the organization ensured adherence to regulations such as GDPR and HIPAA. This compliance readiness reduced the risk of legal penalties and enhanced the organization's reputation for data protection.



DOI: https://doi.org/10.54216/JCIM.160114
Received: November 03, 2024 Revised: January 17, 2025 Accepted: February 12, 2025

Moreover, the proactive security integration during the design phase resulted in cost and time efficiencies. By identifying potential security issues early, the organization avoided expensive rework and delays that often occur when vulnerabilities are discovered in later stages. This approach aligns with best practices that advocate for "shifting left" in security—addressing security concerns as early as possible in the SDLC.

Implementing Database Vault during the design phase directly addresses the gaps identified in our survey, where only 32% of participants recommended this phase for assessing security risks. By prioritizing security at this stage, the organization set a strong foundation for secure development practices throughout the project lifecycle. This strategy mitigated risks and fostered a culture of security awareness among all team members.

### C. Development Phase during SDLC

A code block was essential in the development phase to ensure secure coding practices. This helps developers write secure code by providing mechanisms to prevent SQL injection, cross-site scripting (XSS), and other common vulnerabilities. Code Block integrated seamlessly with the development environment, offering real-time feedback and automated code analysis to identify potential security issues as code was written [4, 15].

Regular code reviews and automated security testing were conducted to ensure adherence to security standards. The development team employed static and dynamic analysis tools to scan for vulnerabilities continuously. This practice caught security flaws early and promoted a culture of quality and accountability within the team. As a result, there was a noticeable reduction in security vulnerabilities in the codebase, leading to more stable and secure software releases [4].

Integrating secure coding practices early in development significantly reduced the need for extensive post-development security patches. By addressing security concerns during the coding phase, the organization maintained consistent performance levels and avoided the technical debt associated with retrofitting security measures after deployment. This proactive approach enhanced the overall security posture of the application and streamlined the development lifecycle [23, 15].

Embedding security into the development lifecycle ensured that it was not an afterthought but a fundamental aspect of the development process. The organization adopted a "security by design" philosophy, where developers were encouraged to consider security implications in every line of code they wrote. This approach included comprehensive training programs for developers on secure coding practices, threat modeling, and familiarity with common attack vectors. Ensuring that developers were aware of and able to implement the necessary security measures empowered them to take ownership of the application's security [23].

Additionally, the development team collaborated closely with security experts to stay updated on emerging threats and evolving best practices. They participated in regular workshops and knowledge-sharing sessions, which helped foster a security-centric mindset across the team. This collaboration also facilitated the integration of security requirements into agile development processes, ensuring that security considerations were included in user stories and acceptance criteria [15].

The use of Code Block and the emphasis on secure coding directly addressed challenges identified in the survey, where many participants indicated the need for better tools and practices during development. By investing in developer education and incorporating advanced security tools, the organization bridged gaps in expertise and resistance to change. This alignment with industry best practices improved code quality and enhanced compliance with regulatory standards such as GDPR and HIPAA [4, 23].

Moreover, the organization established metrics to measure the effectiveness of secure coding initiatives. Key performance indicators (KPIs) included the number of vulnerabilities detected during code reviews, the time taken to remediate security issues, and the percentage of code covered by automated tests. Tracking these metrics provided valuable insights into the progress of security efforts and highlighted areas for continuous improvement.

In conclusion, the organization significantly strengthened its security framework by embedding secure coding practices and utilizing tools like Code Block during the Development phase. This strategy minimized vulnerabilities, reduced long-term maintenance costs, and ensured the final product met security and performance expectations.

### D. Testing Phase during SDLC

In the Testing phase, Real Application Testing (RAT) played a crucial role in the comprehensive security evaluation of the system. RAT enables dynamic analysis, penetration testing, and fuzz testing, ensuring the application can withstand potential threats and perform reliably under various conditions [10, 15]. By simulating real-world attack scenarios, RAT helps identify vulnerabilities that may not be detected through static analysis alone.



DOI: https://doi.org/10.54216/JCIM.160114
Received: November 03, 2024 Revised: January 17, 2025 Accepted: February 12, 2025

Implementing RAT markedly improved query execution time, as performance bottlenecks were identified and optimized during testing. This enhancement improved system efficiency and reduced the attack surface by eliminating delays malicious actors could exploit. By thoroughly testing the application in an environment that closely mimics production settings, the organization could identify and rectify security vulnerabilities before deployment, thereby enhancing the overall reliability and security of the system [10].

Additionally, the organization integrated Continuous Integration (CI) pipelines to automate the testing process. By incorporating security tests into the CI pipeline, developers received real-time feedback on the application's security posture, allowing for quick remediation of any identified issues [11, 25]. This automation ensured that security testing became integral to the development workflow, reducing the likelihood of vulnerabilities slipping through to the production environment.

CI pipelines facilitated Continuous Testing, where automated tests are executed every time new code is committed to the repository. This practice ensured that any changes to the codebase did not introduce new security flaws, maintaining the integrity of the application throughout the development lifecycle. Integrating RAT with CI/CD processes streamlined the testing phase, making it more efficient and effective in catching potential security issues early [11].

Moreover, adopting automated testing tools addressed challenges highlighted in the survey, where 54% of participants reported relying on manual test case design. By shifting to automated and model-based testing approaches, the organization reduced the reliance on manual processes, which are often time-consuming and prone to human error. This transition improved testing efficiency and allowed the testing team to focus on more complex scenarios that require human judgment and expertise [13].

The proactive approach in the Testing phase ensured that security was treated as a critical component of software quality assurance rather than an afterthought. By leveraging advanced testing tools and continuous integration practices, the organization significantly enhanced its ability to effectively detect and mitigate security risks. This strategy not only improved the overall security posture of the application but also ensured compliance with industry standards and regulatory requirements [10, 25].

### E. Deployment Phase during SDLC

During the Deployment phase, Transparent Data Encryption (TDE) was implemented to encrypt data at rest, ensuring that sensitive information remained protected even if unauthorized users accessed the physical media. TDE automatically encrypts the database, associated backups, and transaction log files at the file level without requiring changes to the application layer [8, 13]. This seamless integration meant the encryption process was transparent to applications and users, eliminating the need for code modifications.

The implementation of TDE resulted in a significant increase in transaction throughput, demonstrating that encryption could be managed efficiently without substantial performance degradation. Benchmarking tests conducted post-implementation showed that the system maintained high levels of responsiveness and scalability, debunking common misconceptions that encryption severely hampers performance. This balance between security and efficiency was crucial for preserving user satisfaction and operational effectiveness.

TDE also helped the organization fully comply with data protection regulations like the General Data Protection Regulation (GDPR) and the Health Insurance Portability and Accountability Act (HIPAA). The organization mitigated risks associated with data breaches and unauthorized access by ensuring that data remained secure during and after deployment. Compliance audits reflected the effectiveness of TDE in meeting stringent regulatory requirements, thereby reducing the likelihood of legal penalties and enhancing the organization's reputation for data stewardship [8, 5].

This phase involved rigorous security assessments and configuration reviews to ensure all security measures were correctly applied. The deployment team collaborated closely with security experts to validate encryption settings, key management practices, and access controls. Automated tools were utilized to scan for configuration vulnerabilities, and manual reviews were conducted to verify compliance with security policies. These efforts ensured that encryption keys were securely stored and managed and that only authorized personnel had access to critical encryption operations.

By encrypting data at rest, the organization ensured that even if the physical storage media were compromised—through theft, loss, or unauthorized access—the data would remain unreadable and useless to unauthorized parties. This added layer of security was significant for protecting against insider threats and physical breaches that were not prevented by network security measures alone. TDE's ability to safeguard backups and archives extended this protection to all copies of the data, not just the active database.

Moreover, the deployment phase included the implementation of secure communication protocols to protect data in transit. Protocols such as Transport Layer Security (TLS) were enforced to encrypt data transmitted between



DOI: https://doi.org/10.54216/JCIM.160114
Received: November 03, 2024 Revised: January 17, 2025 Accepted: February 12, 2025

the database and client applications. This comprehensive encryption strategy ensured end-to-end data protection, covering data at rest and in motion.

The organization also established robust monitoring and alerting mechanisms to promptly detect and respond to potential security incidents. Tools were configured to generate alerts for unusual activities, such as unauthorized access attempts or anomalies in data access patterns. Incident response plans were developed and tested to ensure readiness during a security breach.

The proactive measures taken during the Deployment phase directly addressed challenges identified in the survey, where participants emphasized the importance of encryption and highlighted concerns about integration complexities. The organization-alleviated fears about potential negative impacts on system efficiency by demonstrating that TDE could be implemented without significant performance trade-offs. Additionally, the successful deployment of TDE displayed how modern encryption solutions could be integrated smoothly into existing infrastructures, even those involving legacy systems.

The Deployment phase's focus on encryption and rigorous security validation established a strong defence against data breaches, contributing significantly to the overall security framework of the SDLC. It underscored the importance of deploying applications securely, with all necessary protections in place, rather than relying solely on post-deployment security measures.

### F. Maintenance Phase during SDLC

The Maintenance phase involves ongoing monitoring, updates, and improvements to address new security threats and vulnerabilities. During this phase, several key security tools and practices are essential to ensure the database remains secure.

One critical tool is using Password Check utilities to enforce strong password policies consistently. These tools ensure users adhere to security best practices by enforcing rules such as password complexity, expiration periods, and mandatory periodic changes. Regular password audits help prevent unauthorized access due to weak or compromised credentials, a common vulnerability exploited in cyberattacks [4, 13].

Data Masking is another vital practice during the Maintenance phase. It ensures that any sensitive data used in non-production environments, such as testing or development, is anonymized or obfuscated. By masking sensitive information, organizations reduce the risk of data exposure in environments that may not have the same security controls as production systems—implementing data masking results in full compliance with regulations like GDPR and HIPAA, which mandate the protection of personal and sensitive information. This practice effectively protects data during testing and development activities [17].

Data Audit tools are also employed to track and record database activities comprehensively. These tools provide detailed logs of who accessed what data, when, and how. Enhanced visibility into database operations significantly reduces suspicious activities, as unusual patterns can be quickly identified and investigated. Regular security audits and vulnerability assessments are conducted using these logs to maintain a secure environment. Continuous monitoring and proactive security measures ensure the database remains safe and compliant with evolving regulatory requirements [26, 14, 20].

Furthermore, the Maintenance phase includes the application of security patches and updates to database software and related applications. Staying current with patches addresses known vulnerabilities that attackers could exploit. Organizations should establish a routine schedule for reviewing and applying updates, ensuring minimal disruption to operations.

Implementing Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS) can also enhance security during maintenance. These systems monitor network traffic signs of malicious activity and can automatically take action to prevent or mitigate threats. By integrating IDS/IPS solutions with database systems, organizations can respond swiftly to potential security incidents.

Employee Training and Awareness Programs are essential to maintain a strong security posture. Regular training ensures that staff know the latest security threats, understand the importance of data protection, and follow the best practices in their daily activities. This human element is crucial, as many security breaches result from social engineering attacks or inadvertent user errors.

By maintaining a proactive approach to security in the Maintenance phase, organizations can adapt to new threats, ensure continuous compliance with regulatory standards, and uphold the integrity and confidentiality of their database systems. This phase is critical for sustaining the security measures implemented in earlier SDLC phases and making necessary adjustments in response to the changing security landscape.



DOI: https://doi.org/10.54216/JCIM.160114
Received: November 03, 2024 Revised: January 17, 2025 Accepted: February 12, 2025

### G. Integration of Security Practices across the SDLC

Ensuring software security throughout its lifecycle requires adopting systematic security practices at every stage of the SDLC. In the Requirements Phase, security needs such as data encryption, access control mechanisms, and identity management are identified and documented. Early identification of these needs allows for incorporating security objectives into the project's scope, ensuring that security considerations are integral to the system from the outset.

The identified security requirements are integrated into the architecture system in the Design Phase. This integration enables the implementation of proper safeguards, such as secure network architectures, data flow controls, and the establishment of security zones. Designing with security in mind helps mitigate risks associated with architectural flaws and sets a solid foundation for secure development.

During the Development Phase, secure coding techniques prevent potential security flaws. Developers follow established coding standards and practices that mitigate common vulnerabilities like SQL injection, cross-site scripting (XSS), and buffer overflows. Code reviews and static analysis tools detect and address security issues early in the coding process, reducing the likelihood of vulnerabilities entering the production environment.

In the Testing Phase, security is assessed through multiple evaluations, including penetration testing, vulnerability scanning, and code analysis. Tools like Real Application Testing (RAT) simulate real-world attack scenarios, ensuring the system is resilient to known threats. This thorough testing process helps identify and rectify security weaknesses before deploying the system, enhancing its reliability and robustness.

Finally, the Maintenance Phase focuses on continuous monitoring and regularly applying security updates to protect against new threats. Data Audit tools are implemented to monitor database activities, providing detailed logs and alerts for any suspicious actions. Regular security assessments and compliance checks are conducted to ensure that the system adapts to evolving security standards and threat landscapes. This ongoing vigilance ensures data protection from new vulnerabilities and maintains the integrity of the system over time [23, 15, 26].

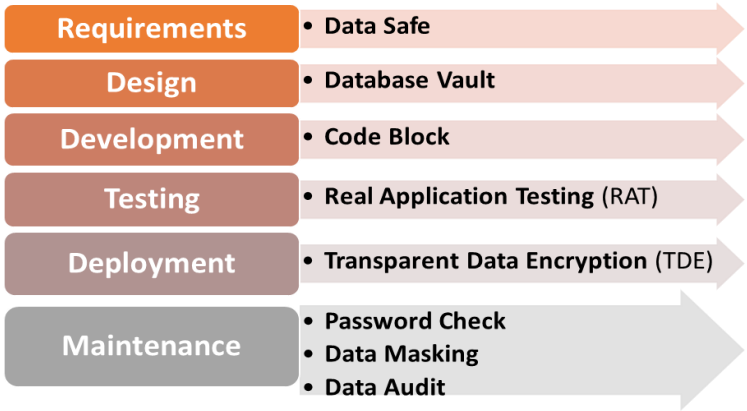


**Figure 19.** Mapping Security Tools to SDLC Phases for Enhanced Database Protection

The mapping of security tools to specific SDLC phases, as shown in Figure 19, highlights the importance of incorporating security practices at every stage of the development cycle. Organizations can significantly reduce the risk of security vulnerabilities by deploying appropriate tools such as Real Application Testing (RAT) during the testing phase and using Data Audit for ongoing maintenance. For instance, RAT allows for dynamic and stress testing of applications under various conditions, identifying potential weaknesses that might not be evident through standard testing methods. Meanwhile, Data Audit tools provide continuous oversight of database activities, enabling prompt detection and response to unauthorized access or anomalies.

This structured approach fosters a proactive security posture; ensuring critical assets are protected throughout the database lifecycle. By integrating security measures into each phase, organizations enhance the overall security of their systems and ensure compliance with regulatory requirements and industry best practices. This comprehensive integration reduces the likelihood of costly security breaches and supports the development of secure, reliable software applications.



DOI: https://doi.org/10.54216/JCIM.160114
Received: November 03, 2024 Revised: January 17, 2025 Accepted: February 12, 2025

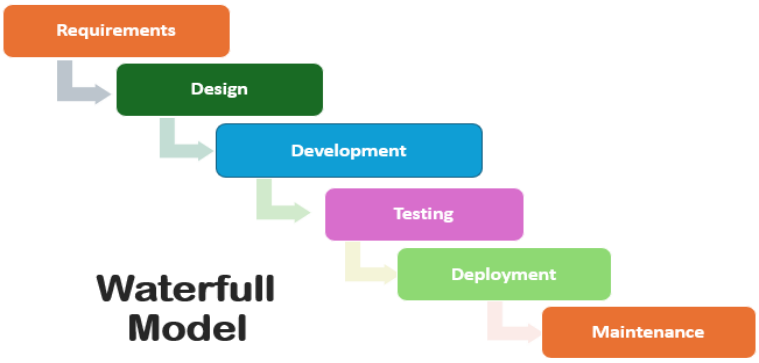


**Figure 20.** Traditional Waterfall Model

Figure 20 illustrates the traditional Waterfall model in the Software Development Life Cycle (SDLC). In this model, each phase is executed sequentially, moving from Requirements to Maintenance without much focus on security integration. Typically, security is only considered in the Testing or Maintenance phases, leading to potential vulnerabilities that could have been addressed earlier. The lack of early-stage security assessment makes it challenging to mitigate risks effectively, and security measures are often reactive, leading to higher costs and complexity later in the cycle.

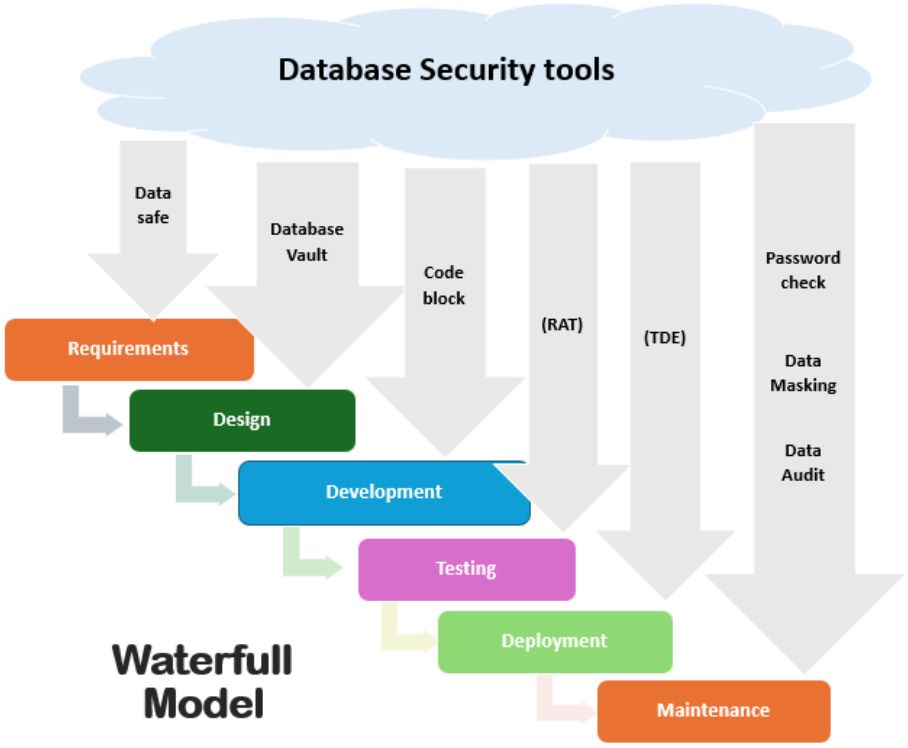


**Figure 21.** Waterfall Model with Enhanced Security Integration

In Figure 21, the Waterfall model is modified by integrating specific security tools and practices into each phase of the SDLC. The proposed framework introduces proactive security measures, starting in the Requirements phase, where security needs like encryption and access control are identified. During the Design phase, the architecture incorporates security solutions such as Database Vault to enforce strict access policies. In Development, secure coding practices are emphasized with tools like Code Block, ensuring that vulnerabilities are identified and fixed as the code is written. Testing becomes more comprehensive with dynamic tools like Real Application Testing (RAT), simulating real-world attacks to identify potential risks. Finally, the Deployment and Maintenance phases include tools like Transparent Data Encryption (TDE) and continuous monitoring with auditing tools to ensure



DOI: https://doi.org/10.54216/JCIM.160114
Received: November 03, 2024 Revised: January 17, 2025 Accepted: February 12, 2025

long-term security. This approach ensures that security is embedded at every stage, reducing risks and ensuring compliance.

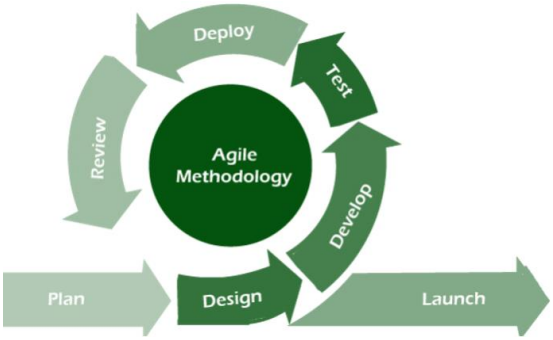


**Figure 22.** Traditional Agile Methodology

Figure 22 illustrates the traditional agile methodology, renowned for its iterative and flexible approach. However, security is frequently deferred until the Testing or Maintenance phases, after the core functionality has been implemented [15]. This postponement of security considerations can result in critical vulnerabilities being identified late in the development cycle, increasing the complexity and cost of remediation. Due to Agile's emphasis on rapid delivery, the focus tends to remain on functionality, often relegating security to a secondary priority in subsequent iterations. This reactive approach can introduce security gaps, exposing the system to potential risks.

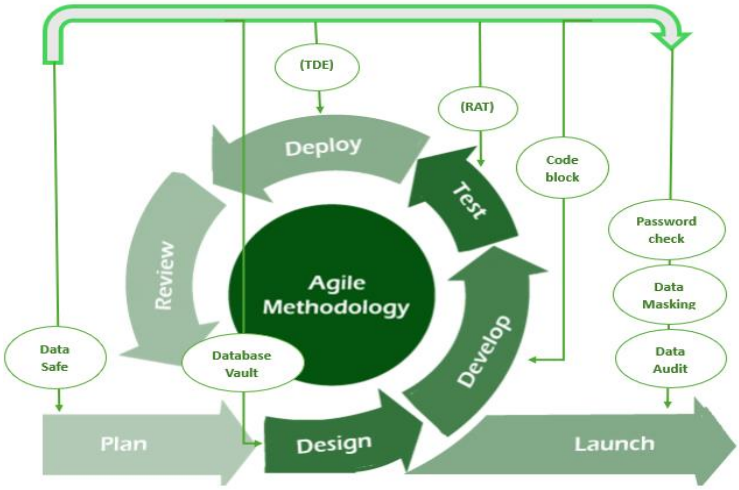


**Figure 23**. Agile Methodology with Integrated Database Security Framework

In Figure 23, the agile methodology is enhanced by integrating security throughout each iterative cycle based on the proposed framework. Security considerations are incorporated from the Planning and Design phases, such as identifying sensitive data and implementing security controls early. During each development sprint, secure coding practices are applied using tools like Code Block to minimize vulnerabilities as code is written. Real-time testing uses tools like RAT to assess and address security risks in each iteration continuously. Security tools like Data Masking and Transparent Data Encryption (TDE) are applied during Deployment, ensuring both data in use and at rest are secure. Finally, ongoing monitoring and audits are performed during the Review and Maintenance phases to ensure the system remains safe even after deployment. This framework ensures that security is not an afterthought but an integral part of the agile process, continuously evolving alongside the application.

Table 3 presents a comparative analysis of key aspects between previous studies and the present work regarding integrating database security practices within the Software Development Life Cycle (SDLC). This table highlights the limitations identified in prior research and how the current study addresses these gaps. By contrasting the approaches, it becomes evident how the present work advances the field by providing a more comprehensive, adaptable and proactive security framework.



DOI: https://doi.org/10.54216/JCIM.160114
Received: November 03, 2024 Revised: January 17, 2025 Accepted: February 12, 2025

**Table 3:** Comparison of Previous Studies and Present Work

| Aspect/Feature | Previous Studies | Present Work |
|---|---|---|
| Focus on All SDLC Phases | No (focused on some phases like testing or risk assessment) | Yes (addressed all phases comprehensively) |
| Coverage of Security Tools | Some tools discussed individually | Yes (integrated all relevant tools within a unified framework) |
| Integration of Security from Requirements Phase | Rarely or never | Yes (integration began from the Requirements phase) |
| Detailed Techniques for Each Phase | Limited or none | Yes (provided detailed techniques for every phase) |
| Addressing Security Gaps with Solutions | Identified gaps but rarely provided solutions | Yes (identified gaps and provided practical solutions) |
| Adaptability to Different Methodologies | Limited to specific methodologies (e.g., Agile only) | Yes (adaptable to various methodologies, including Waterfall) |
| Scalability and Adaptability to Organizations | Limited; not adaptable to all sizes/types of organizations | Yes (framework designed to be scalable and adaptable) |
| Early Integration of Security Practices | Sometimes, but not consistently | Yes (integrated security practices early and consistently) |

In summary, Table 3 illustrates the significant advancements made in the present work compared to previous studies. While earlier research often focused on specific phases of the SDLC or individual security tools without providing comprehensive solutions, the current study addresses these shortcomings by integrating security practices across all SDLC phases. It offers detailed techniques for each phase, ensures early and consistent security integration starting from the requirements phase, and provides practical solutions to identify security gaps. Furthermore, the proposed framework is adaptable to different development methodologies and scalable to organizations of various sizes, making it a versatile tool for enhancing database security in diverse contexts. This comprehensive approach underscores the contribution of the present work in advancing the field of database security within software development.

## 6. Limitations

While this study provides a comprehensive framework for enhancing database security throughout the Software Development Life Cycle (SDLC), several limitations must be acknowledged. Recognizing these limitations is essential for interpreting the findings accurately and guiding future research efforts.

### A. Limitations of the Survey Approach

Firstly, the survey conducted as part of this study was limited in scope and relied on convenience sampling, which may introduce bias. The participants were primarily professionals accessible through our networks, potentially leading to a sample that was not fully representative of the wider industry. This approach may have fevered specific organizations or sectors, affecting the results' generalizability. The focus on entities was intentional to ensure the safety, quality, and privacy of the data collected. However, it may have inadvertently introduced bias toward the practices and challenges prevalent within those specific organizations.

Secondly, the design and methodology of our survey were influenced by the paper titled "Cross-factor analysis of software engineering practices versus practitioner demographics: An exploratory study in Turkey"[27]. While their research provided valuable insights and a foundational structure for our questionnaire, adapting their methods to our context may have limitations. Cultural, regional, and industry-specific factors could affect the applicability of their survey design to our study. Although their methodology enhanced the depth of our survey, it may have introduced constraints related to differences between their study context and our own.



DOI: https://doi.org/10.54216/JCIM.160114
Received: November 03, 2024 Revised: January 17, 2025 Accepted: February 12, 2025

Additionally, the detailed methodology of our survey questions, adapted from Garousi et al. (2015), may not have fully captured the nuances of database security practices across diverse industries and regions. While the original study focused on software engineering practices in Turkey, residual elements might not entirely align with the global context of our research. This could affect the reliability and validity of the data, as some regional practices or challenges might not have been adequately represented.

Moreover, the selective focus on specific entities during the survey, aimed at ensuring data privacy and quality, may have underrepresented some security tools or challenges prevalent in other organizations. This limited focus may particularly affect smaller companies or those operating in different regulatory environments, leading to incomplete insights into database security challenges.

### B. Limitations of the Proposed Solution

While the proposed framework offers a comprehensive approach to integrating security throughout the SDLC, several limitations emerge from the solution itself:

The framework's reliance on multiple security tools (e.g., Data Safe, Database Vault, Real Application Testing) and practices across every SDLC phase can be resource intensive. Small or medium-sized enterprises (SMEs) may struggle to allocate the necessary resources—in terms of staff, budget, and infrastructure—needed for effective implementation. This complexity can also lead to longer development timelines, especially for teams unfamiliar with advanced security tools.

Some organizations may rely on legacy systems incompatible with modern security tools such as Transparent Data Encryption (TDE) or Database Vault. Integrating these security technologies without disrupting existing workflows requires expertise and meticulous planning, which can pose a significant barrier to adoption.

Maintaining the security framework requires continuous monitoring, regular audits, and updates to respond to evolving threats. However, resource constraints or lack of skilled personnel may hinder organizations from applying updates consistently, increasing the risk of vulnerabilities over time.

While the framework aims to be adaptable, its effectiveness may diminish in large-scale or highly distributed environments, such as multi-cloud systems or cross-border infrastructures. Scaling the solution without compromising security or performance can be challenging, and additional customizations might be required to meet specific needs.

Implementing the proposed solution depends heavily on employee training, collaboration, and management support. The framework's effectiveness may be limited if staff are not adequately trained, or the organization lacks a security-conscious culture. Human errors or lack of adherence to security policies could lead to vulnerabilities, regardless of the technical controls.

Some security measures like encryption and real-time monitoring may introduce performance overhead. Organizations must carefully balance security needs with system performance, particularly in high-transaction environments. Inadequate optimization of security tools may result in delays or disruptions in critical operations.

### C. Temporal Relevance of Findings

The dynamic nature of cybersecurity threats and rapid advancements in database technologies presents another limitation. The data collected reflects practices and challenges during the survey, but new threats or technologies may have emerged since then. As such, the findings offer a snapshot rather than a long-term solution, necessitating continuous updates to the security framework and practices.

### D. Contextual Constraints of the Framework

The implementation and impact analysis of the proposed framework were conducted within specific organizational contexts. Depending on the organization's resources, expertise, and security infrastructure, the results may vary when applied to different environments. Organizational culture, employee training levels, and management support play critical roles in successfully adopting the framework. Therefore, while the framework is designed to be adaptable, its effectiveness may be influenced by these contextual factors.

## 7. Conclusions and future work

This study addressed integrating comprehensive database security practices across all phases of the Software Development Life Cycle (SDLC). The problem lies in the industry's fragmented security approaches, where security measures are applied only at later stages, leaving systems vulnerable to threats during development and deployment. Moreover, existing research does not provide a detailed, structured framework for aligning database security tools with each SDLC phase, highlighting a significant gap in the literature.



DOI: https://doi.org/10.54216/JCIM.160114
Received: November 03, 2024 Revised: January 17, 2025 Accepted: February 12, 2025

To bridge this gap, our research proposed a framework that embeds security tools and practices throughout the SDLC, from planning and requirements gathering to deployment and maintenance. This approach ensures that security considerations are incorporated early, reducing the risk of vulnerabilities and increasing the overall security posture of systems. We identified key tools such as Data Safe, Database Vault, and Transparent Data Encryption (TDE) through a survey involving IT professionals and database administrators. We mapped them to specific phases of the SDLC. This mapping provides a practical guide for organizations seeking to implement secure software development practices.

Our findings demonstrate that integrating security tools early in the SDLC offers several advantages over previous reactive security approaches. For instance, applying tools like Data Safe during the planning phase enhances regulatory compliance from the outset, while real-time monitoring with auditing tools strengthens system resilience throughout the lifecycle. The framework also emphasizes secure coding practices, continuous testing, and post-deployment monitoring, addressing common challenges reported by survey participants, such as high costs and integration with legacy systems.

However, the proposed solution is not without limitations. Implementing the framework requires significant resources, including skilled personnel and advanced tools, which may present challenges for smaller organizations. Additionally, integrating security tools across distributed infrastructures, such as multi-cloud environments, may require further customization. Balancing the need for security with system performance also remains a key consideration, particularly for high-transaction systems.

In future work, further research is needed to evaluate the framework's effectiveness across different industries and development methodologies, such as Agile and DevOps. Another promising area is exploring emerging technologies like artificial intelligence and machine learning to enhance threat detection and response. Additionally, conducting longitudinal studies to assess the long-term impact of early security integration will provide deeper insights into the sustainability of the proposed framework. Expanding the scope of research to include different regions and sectors will also help capture a more comprehensive understanding of database security practices globally.

Future studies can contribute to developing more robust, adaptable security practices by addressing the identified challenges and refining the framework over time. Ultimately, this research is a foundation for organizations seeking to enhance their database security and align their development processes with evolving industry standards and regulatory requirements.

**Funding:** “This research received no external funding”

**Conflicts of Interest:** “The authors declare no conflict of interest.”

**References**

[1] J. Smith and T. Johnson, "Data security governance in the era of big data: Status, challenges, and solutions," *Journal of Big Data*, 2021.

[2] O. E. Olorunshola and F. N. Ogwueleka, "Review of System Development Life Cycle (SDLC) Models for Effective Application Delivery," in *Proc. Information and Communication Technology for Competitive Strategies (ICTCS 2020)*, Lecture Notes in Networks and Systems, vol. 191, Springer Nature, 2022, pp. 10-20. doi: 10.1007/978-981-16-0739-4_2.

[3] W. W. Royce, "Managing the Development of Large Software Systems: Concepts and Techniques," in *Proc. IEEE WESCON*, 1970.

[4] S. Davis and J. Harris, "Mitigating Database Security Threats: A Guide to Best Practices," *Information Systems Journal*, vol. 35, no. 1, pp. 75-89, 2020.

[5] J. Smith and M. Garcia, "Advanced Techniques in Data Encryption for Secure Cloud Storage," *Computers & Security*, vol. 92, p. 101113, 2020.

[6] D. Burdick and C. Newman, "Banking and Finance Data Breach: Costs, Risks and More," *Security Intelligence*, 2021.

[7] H. Fisher and B. White, "Effective Risk Management Strategies for IT Security," *Journal of Information Security*, vol. 11, no. 4, pp. 301-317, 2020.

[8] E. Peters and R. Adams, "Advanced Encryption Standards and Their Applications in Modern Databases," *Journal of Computer Security*, vol. 29, no. 1, pp. 47-65, 2021.



DOI: https://doi.org/10.54216/JCIM.160114
Received: November 03, 2024 Revised: January 17, 2025 Accepted: February 12, 2025

[9] D. Bărbulescu, A.-C. Enache-Ducoffe, and M. Togan, "A new comparative study of database security," *Romanian Journal of Information Technology and Automatic Control*, vol. 33, no. 3, pp. 17-28, 2023. doi: 10.33436/v33i3y202302.

[10] A. Miller and G. Thompson, "Innovations in Database Security Technologies: A Comprehensive Overview," *Journal of Information Technology*, vol. 42, no. 5, pp. 223-237, 2020.

[11] Stackify, "Security in Software Development: Best Practices," 2021. [Online]. Available: https://stackify.com/security-in-software-development-best-practices/.

[12] R. A. Teimoor, "A Review of Database Security Concepts, Risks, and Problems," *UHD Journal of Science and Technology*, vol. 5, no. 2, pp. 38-46, 2021. doi: 10.21928/uhdjst.v5n2y2021.pp38-46.

[13] R. M. Green and J. T. Boys, "Implementation of Pulsewidth Modulated Inverter Modulation Strategies," *IEEE Trans. Ind. Appl.*, vol. IA-18, no. 2, pp. 138-145, Mar. 1982. doi: 10.1109/TIA.1982.4504048.

[14] Oracle Corporation, "Oracle Data Safe: Comprehensive Security for Your Oracle Databases," Oracle White Paper, 2021. [Online]. Available: https://www.oracle.com/security/data-safe/.

[15] L. Anderson and J. White, "Embedding Security in Agile Development Processes," *Journal of Systems and Software*, vol. 165, p. 1107, 2020.

[16] S. Sharma, "A Comparative Study on Database Breach and Security in Contemporary Perspective," *Int. J. Res. Appl. Sci. Eng. Technol. (IJRASET)*, vol. 12, no. 8, 2024.

[17] M. B. Mousa, H. A. Kholidy, and M. Rasslan, "A comprehensive survey of data masking techniques," *Journal of Information Security and Applications*, vol. 55, p. 102578, 2020.

[18] A.-M. Stanciu and H. Ciocârlie, "Integrating Security into the Software Development Life Cycle: A Systematic Approach," in *Proc. 2023 Int. Conf. Electron. Electr. Eng. Comput. Sci. (ICEECET)*, 2023, pp. 1-6. doi: 10.1109/ICECET58911.2023.10389547.

[19] V. Casola, A. De Benedictis, C. Mazzocca, and V. Orbinato, "Secure software development and testing: A model-based methodology," *Computers & Security*, 2023. doi: 10.1016/j.cose.2023.103639.

[20] Oracle Corporation, "Oracle Data Safe: User Guide," Oracle Help Center, 2020. [Online]. Available: https://docs.oracle.com/en/cloud/paas/data-safe/udscs/index.html.

[21] M. Alenezi and S. Almuairfi, "Security Risks in the Software Development Lifecycle," *Int. J. Recent Technol. Eng. (IJRTE)*, vol. 8, no. 3, pp. 7048-7055, 2019. doi: 10.35940/ijrte.C5374.098319.

[22] R. A. Khan, S. U. Khan, M. A. Akbar, and M. Alzahrani, "Security risks of global software development life cycle: Industry practitioner's perspective," *Journal of Software: Evolution and Process*, 2022. doi: 10.1002/smr.2521.

[23] Y. Valdés-Rodríguez, J. Hochstetter-Diez, J. Díaz-Arancibia, and R. Cadena-Martínez, "Towards Integrating Security Practices in Agile Software Development: A Systematic Mapping Review," *Applied Sciences*, vol. 13, no. 7, p. 4578, 2023. doi: 10.3390/app13074578.

[24] Y. Mothanna, W. ElMedany, M. Hammad, R. Ksantini, and M. S. Sharif, "Adopting security practices in software development process: Security testing framework for sustainable smart cities," *Computers & Security*, vol. 144, p. 103985, 2024.

[25] Red Hat, "Security in the Software Development Lifecycle (SDLC)," 2021. [Online]. Available: https://www.redhat.com/en/topics/security/security-in-the-software-development-lifecycle.

[26] K. Williams and R. Smith, "Implementing Advanced Security Measures in Database Systems: A Case Study," *Journal of Database Security*, vol. 12, no. 1, pp. 34-49, 2021.

[27] V. Garousi, A. Coşkunçay, A. Betin-Can, and O. Demirörs, "Cross-factor analysis of software engineering practices versus practitioner demographics: An exploratory study in Turkey," *Journal of Systems and Software*, vol. 111, pp. 108-129, 2015. doi: 10.1016/j.jss.2015.09.013.



DOI: https://doi.org/10.54216/JCIM.160114
Received: November 03, 2024 Revised: January 17, 2025 Accepted: February 12, 2025