International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 13 Issue: 04 | Apr 2026 www.irjet.net p-ISSN: 2395-0072

Software Development Life Cycle: Adopting a Security-First Approach

Bulupiy Galati Joel

Computer Science Faculty, Dept. Network & Security, Université Protestante au Congo, Democratic Republic of The Congo

---

Abstract – *The rapid evolution of software development has intensified cybersecurity challenges across web application, embedded systems, distributed architectures, Internet of Things (IoT) and artificial intelligence solution. In traditional approaches, security is often treated as a final stage in the Software Development Life Cycle (SDLC), resulting in increased vulnerability risks, higher remediation costs, delayed deployments, and potential regulatory penalties. This paper advocates a Security-First approach, which integrates security practices form the initial phases of the SDLC rather than as a reactive measure. It examines major SDLC methodologies, evaluation their characteristics, use cases, security integration potential and optimal application contexts. Drawing on influential standards such as the NIST Secure Software Development Framework (SSDF) and Secure SDLC (SSDLC), the study highlights the benefits of proactive security integration. Results indicate that adopting a Security-First mindset significantly reduces risks while preserving system performances and availability.*

Key Words: **Software Development Life Cycle (SDLC), Security-First, DevSecOps, Secure SDLC, NIST SSDF, Threat Modeling, CIA Triad**

1.INTRODUCTION

Over the past three decades, software development has advanced at an extraordinary pace. This rapid growth requires a methodical and proactive approach to securing the entire information system while ensuring proper functionality and availability.

In light of emerging daily threats, it is essential to adopt a Security-First methodology. This approach promotes incorporating security layers from the beginning of the software development process, rather than threating security as a final security or corrective measure after a technical failure or an attack.

Practically, many traditional SDLC models do not integrate security in detailed and systematic manner. To address this gap, the NIST Secure Software Development Framework (SSDF) is recommended as a robust method for embedding security practices across every phase of the SDLC [1].

To implement this strategy effectively the Secure SDLC (SSDLC) is employed. Unlike classical methods that defer security and testing to the end of the cycle, SSDLC incorporates security principles, architectural decisions, tools, and procedures from the outset [2]. Building secure foundations from the starts is vital for preventing incidents, this approach applies to diverse projects, including web applications, embedded software, disturbed systems, network programming, Internet of Things (IOT), and artificial intelligence systems.

2. SDLC Methodologies

Several SDLC methodologies exist. This section analyses each according to four criteria: characteristics, use cases, security integration and recommended application contexts. The comparison is summarized in the table below.

Table -1: Comparison of SDLC Methodologies with Security-First Integration

Methodology	Characteristics	Use Cases	Security Integration (Security-First/DevSecOPs)	When to use
Waterfall	Linear and sequential; each phase must be completed before the next.	Projects with stable require; nets and well-defined scope.	Security typically added late; difficult to retrofit changes	Simple projects with minimal changes.
Iterative	Progressive development thought repeated cycles.	Projects requiring user feedback and moderate requirement evolution.	Security integrated in each iteration (threat modelling and testing per cycle)	Medium-sized projects with gradual changes.
Agile	Combine incremental and iterative approaches with short sprints.	Dynamics projects, start-ups, mobile and web applications with changing	Highly effective with DevSecOps; security incorporated in every sprint	Fast-paced environments needing rapid delivery

		requirements		
V-Model	Waterfall extension with parallel verification and validation	Safety-critical, suitable for finance, aerospace	Security testing planned in parallel with each development phase	Projects requiring rigorous validation and stable requirements
RAD (Rapid Application Development)	Rapid prototyping with intensive use feedback	Small projects or MVPs needing quick production deployment	Basic security controls included from early prototypes [3]	Small-scale projects with tight deadlines and strong user involvement
Spiral	Iterative with strong risk analysis in each planning from planning, risk, engineering to evaluation	High-risk and complex projects, like embedded systems, military	Early threat modeling and risk mitigation in every spiral [4]	Projects with high technical or security risks
DevOps	Integration of Development and Operations with CI/CD and automation [5]	Cloud-native applications, microservices, and frequent deployment	Excellent for DevSecOps: automated security scanning, continuous monitoring	Cloud environments requiring speed, reliability, and continuous security
Incremental	Development by functional modules delivered sequentially	Large projects divisible in to deliverable modules	Security testing applied to each module	Large systems where partial early delivery is needed

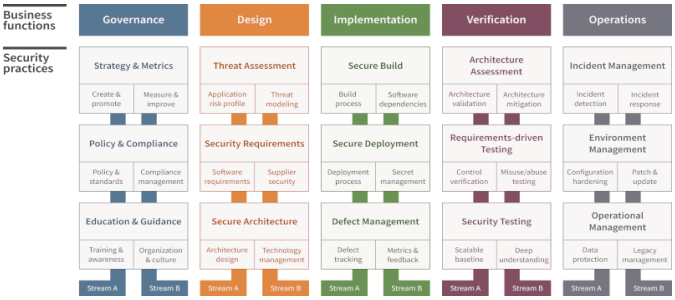

Fig -1: Software Assurance Maturity Model by OWASP

Security must also be addressed through standards and norms, not only as a technical concern, a common mistake made by many organizations during software development. Enterprises should integrate security across all dimensions such as technical, organizational and normative from project initiation [6] [7].

This leads to the adoption of SSDLC which embeds necessary security requirements into every phase of development process, from design trough deployment and maintenance [8].

2. 1 Risk of not adopting security first

No software or information system in the world, regardless of its sophistication, is 100% immune to attacks. This reality has popularized the Zero Trust concept, which is based on the principle of « Never Trust, always verify." It assumes that threats may exist both inside and outside the traditional security perimeter [9].

Appropriate security layers must be implemented to ensure better performance and high availability with a target of 99.9% uptime. However, excessive security mechanisms can overburden the software, increase resource consumption, complicate maintenance and control, and ultimately hinder its proper functioning and use experience.

A Security-First approach, guided by frameworks such as the NIST SSDF, seeks an optimal balance: proactive security integration from the earliest stages while eliminating unnecessary complex configurations [10].

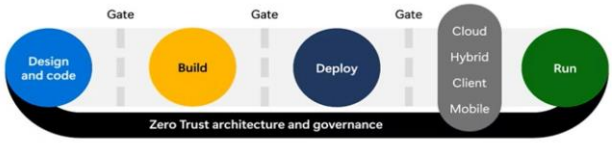

Fig -2: Zero Trust Workflow: Security First by Microsoft

This figure illustrates the concept effectively. Security controls are applied at each stage and component of the system, with verification required before moving from one stage to another. Each zone is separated by control gates.

2. 2 Security Layers in the SDLC

In the SDLC, even when emphasizing a Security-First approach, it is essential to ensure the availability of the entire information system, including during attacks. The system must remain resilient. Therefore, rigorous risk management and the handling of potential failures are strongly recommended through the development cycle.

Technologically Security rests on the CIA Triad (Confidentiality, Integrity, Availability):

- **Confidentiality**: Protects data from unauthorised access. Achieving a high protection level, close to 99.9%, strengthens user confidence in the security of their personal data. This requirement is supported by international standards such as ISO 270001, which defines requirements for an Information Security Management System and PCI DSS (Payment Card Industry Data Security Standard), particularly relevant for systems handling payment card data and baking transactions.
- **Integrity**: Ensures the accuracy, truthfulness, and non-alteration of data throughout its lifecycle in the system. It prevents unauthorised of fraudulent modifications.
- **Availability**: Guarantees that authorized users can access information and services when needed, even under stress or in the presence of attacks.

A major risk to prevent is **the Man-in-the-Middle** (MITM) attack. In this technique, as attacker secretly position themselves between two parties without their knowledge. Acting as a proxy, the attacker can intercept, eavesdrop on or modify communications in Realtime. This can lead to massive data leaks or alteration of exchanges, directly compromising he confidentiality and integrity of data flowing through the system [11].

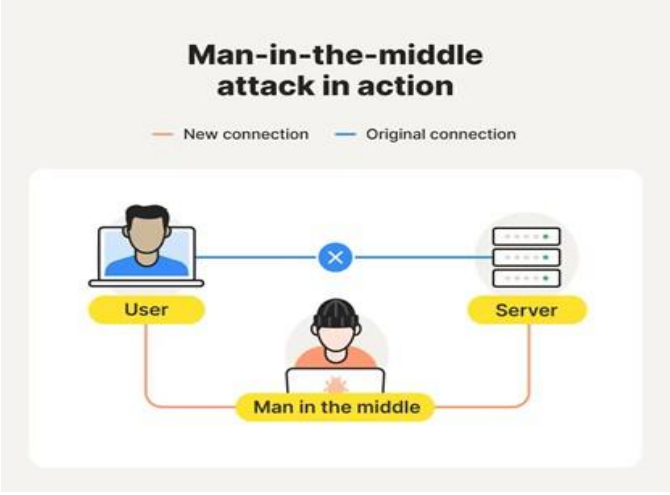

Fig -3: MITM Attack Illustration

As shown in the figure, we have three actors involved and behaving as expected in the table below:

Table -2: Actors actions in MITM Attack

Actor	Action
User	Send requests to the server to retrieve, create, or modify information
Server	Stores and processes information may trigger events based in configurations
Man-in-the-Middle	Uninvited actor who uses identity spoofing to stead or manipulates data transiting

To encounter these threats, the Security-First approach recommends integration appropriate controls, such as encryption and authentication from the design phase and across all layers of the SDLC.

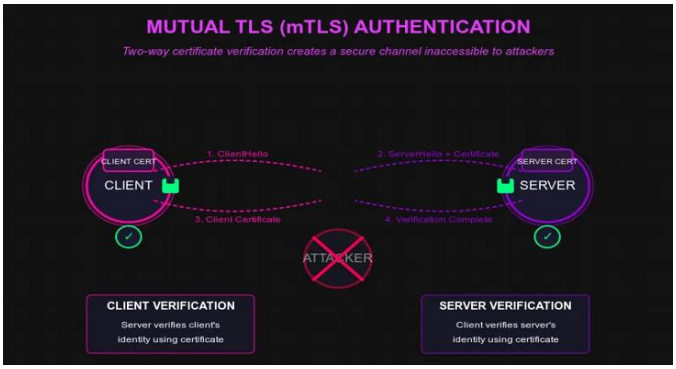

Fig -4: Mutual Authentication Diagram

As illustrated in the figure security is enforced on both side trough certificate verification by each action before proceeding with deeper operations. This provides additional protecting for data in transit [12].

3. Security-First Implementation in the Traditional SDLC Phases

The traditional SDLC consists of seven well-defined phases. A Security-First approach, informed by the NIST SSDF and the foundational principles outlined by Gary McGraw's, requires that security activities be embedded in each phase rather than added as an afterthought. The following paragraphs describe each phase and the corresponding Security-First practices.

Fig -5: SDLC Phases

1. **Planning**: conduct initial risk assessments and define high-level security requirements in alignment with business objectives and compliance obligations such as ISO 27001. Identify potential threats and allocate resources for security activities from the outset.

2. **Requirements Analysis**: specify both functional and non-functional security requirements using the CIA Triad. Perform preliminary threat modeling to capture needs early and ensure they are documented as part of the requirements baseline.

3. **Design**: apply secure architecture principles and conduct detailed threat modeling, by using methodologies like STRIDE (Spoofing, Tampering, Repudiation, Information Disclosure, Elevation of Privilege). Design for least privilege, secure defaults, and defence-in-depth to eliminate vulnerabilities at the architectural level [13].

4. **Development**: follow secure coding standards, perform static application security testing (SAST), and conduct peer code reviews with a security focus. Automated tools are used to detect common vulnerabilities before they reach later stages.

5. **Testing**: integrate dynamic application security testing (DAST), penetration testing, and fuzz testing. Verify that security controls function correctly and do not introduce unacceptable performance degradation.

6. **Deployment**: implement secure configuration management, infrastructure-as-code scanning, and hardened CI/CD pipelines. Conduct a final security review and apply all necessary hardening measures before release.

7. **Maintenance & Support**: establish continuous monitoring, vulnerability management, timely pathing, and incident response capabilities. Risks are regularly reassessed and security controls updated as the system evolves or new threats emerge.

This structured integration, supported by the NIST SSDF and McGraw's touch point framework, shifts security left in the development process [14], reduces long-term remediation costs, and significantly enhances overall system resilience.

3. CONCLUSIONS

Adoption of a Security-First approach shifts security from a reactive constraint to a proactive strength integrated into the development process. By combining suitable SDLC methodologies with frameworks like NIST SSDF and respecting the CIA Triad, organization scan significantly reduces risks while maintaining performance and availability. We are aiming to improve future work to include AI-driven security in DevSecOps pipelines, and also Network Security.

REFERENCES

[1] S. Murugiah, S. Karen and D. Donna, "Secure Software Development Framework (SSDF)," NIST Special Publication 800-218, 2022.

[2] A. Mustyala, "Security-First DevOps: Best Practices for Safeguarding Continuous Delivery Pipelines," *ISAR Journal of Multidisciplinary Research and Studies,* vol. 1, no. 4, p. 3, 2023.

[3] G. Aradhyula, "The security-first agile playbook: Embedding DevSecOps into program management," *World Journal of Advanced Engineering Technology and Sciences,* vol. 16, no. 03, p. 17, 2025.

[4] OWASP, "Dev Guide Owasp," Owasp, [Online]. Available: https://devguide.owasp.org/en/02-foundations/02-secure-development/. [Accessed 28 03 2026].

[5] J. Twist, "Zuplo," 05 March 2025. [Online]. Available: https://zuplo.com/learning-center/mitm-attack-prevention-guide. [Accessed 28 March 2026].

[6] G. McGraw, Software Security: Building Security In, Addison-Wesley, 2006.

[7] N. Davis, Secure Software Development Life Cycle Processes: A Technology Scouting Report, Carnegie Mellon University Software Engineering Institute, 2005.

[8] JFrog, "JFrog," JFrog, [Online]. Available: https://jfrog.com/learn/devsecops/ssdlc-secure-

software-development-lifecycle/. [Accessed 28 March 2026].

[9] OWASP, "OWASP," OWASP, 2022. [Online]. Available: https://owasp.org/www-project-samm/. [Accessed 28 March 2026].

[10] P. Belagatti, "The New Stack," 29 10 2021. [Online]. Available: https://thenewstack.io/zero-trust-security-and-the-software-development-lifecycle. [Accessed 28 March 2026].

[11] Microsoft, "Microsoft," [Online]. Available: https://www.microsoft.com/en-us/securityengineering/sdl/practices. [Accessed 28 March 2026].

[12] M. Birchall, "Us Norton," Norton, 26 03 2020. [Online]. Available: https://us.norton.com/blog/wifi/what-is-a-man-in-the-middle-attack. [Accessed 28 March 2026].

[13] L. Microsof, "Learn Microsoft," Microsoft, 19 03 2026. [Online]. Available: https://learn.microsoft.com/en-us/azure/well-architected/security/secure-development-lifecycle. [Accessed 28 March 2026].

[14] D. Puzas, "New Relic," New Relic, 23 05 2025. [Online]. Available: https://newrelic.com/blog/security/how-to-leverage-security-in-your-software-development-lifecycle. [Accessed 28 March 2026].

BIOGRAPHIES

Software Engineer, Bachelor's Degree in Software Engineering from Université Protestante au Congo
Researcher in Computer Science
Master's Student specializing in Cybersecurity, Network and Systems