Telefónica Tech

Building a secure Future

Deploying Secure Software Development Life Cycle

What is SSDLC?

SSDLC (Secure Software Development Life Cycle) is a software development approach that integrates vulnerability identification and mitigation throughout the process. It aims to optimize workflows through specific activities, allowing teams to reduce risks and create more resilient applications in the face of constantly evolving threats.

Unlike **SDLC**, which structures and controls the development process, **SSDLC** follows the same phases, but with security activities integrated into each one.

What are the phases of the SSDLC *implementation*?

Planning and Analysis
Identification and assessment of security risks, through the creation of a threat model and the selection of security requirements.

Design
Integration of safety measures into the design, as well as security testing for each phase.

Implementation
Following secure coding guidelines, using analysis tools to identify and correct vulnerabilities during development.

Testing
Performing static and dynamic security testing to secure the software prior to deployment and establishing Security Gates.

Maintenance
Continuous monitoring of software in production, periodic penetration testing, and execution of incident response plans to manage emerging risks.

What are the reasons for *implementing* SSDLC?

Security breaches in traditional **SDLC** are severe due to its reactive approach, which only adds testing at the end.

In contrast, **SSDLC** mitigates risks by integrating vulnerability scanning, insecure configurations, and third-party dependencies during development.

Vulnerability Reduction
Integrating security at every stage significantly reduces critical vulnerabilities from the earliest stages.

Cost Savings
Helps companies reduce security breach management and risk mitigation costs.

Decreased Security Incidents
Improving software quality and security increases customer satisfaction.

Increased Customer Satisfaction
Oprimizes efficiency, reduces development times, and improves software quality.

Improved Efficiency
Proactive security measures significantly reduce incidents after software deployment.

What do we bring to SSDLC *development* in your company?

The implementation of **SSDLC** throughout the entire process phase has taught us that:

Early Integration
It is essential to incorporate security from the early stages of development to ensure its effectiveness.

Continuous Training
Investing in continuous training of developers is crucial to keep them up to date on security.

Test Automation
Security test automation, such as **SAST, SCA,** and **DAST,** is key to efficiently identify and mitigate risks.

Interdepartmental Collaboration
Cooperation between development, security, and operations teams is critical to the success of **SSDLC,** giving birth to the figure of **DevSecOps.**

Benefits of *adopting* SSDLC in your company

1. **Initial Risk Assessment**
   Performing a security risk analysis at the beginning of the project.
2. **Threat Planning and Modeling**
   Incorporation of a security model in the planning and identification of threats to define security requirements.
3. **Staff Training**
   Allocation of resources to the continuous training of developers and teams involved in the development.
4. **Use of Security Tools**
   Implementation of **SAST, SCA** and **DAST** tools for vulnerability detection during development and testing.
5. **Continuous Monitoring**
   Establishment of constant monitoring and periodic reviews to maintain software security in the face of new threats.

Summary

Security approach
The **SDLC** focuses on delivering software without prioritizing security, while the **SSDLC** integrates it from the start.

Security Phases
The **SSDLC** incorporates security in each development phase (Threat Modeling, Static Application Security Testing **(SAST)**, Software Composition Analysis **(SCA)**, Dynamic Application Security Testing **(DAST)**, Hardening) while the **SDLC** does not.

Vulnerability Prevention
SSDLC detects and mitigates vulnerabilities during the process, not only after development is complete.