**SDLC and secure coding practices: the ultimate guide for 2024**

As the threat landscape grows and the costs of data breaches increase, organizations are looking to adopt secure software development lifecycle (SDLC) methodologies. Here's how to do it.

As the threat landscape grows and the costs of data breaches increase, organizations are looking to adopt secure software development lifecycle (SDLC) best practices and methodologies. Secure SDLC is a multi-step approach that comprises a set of rules, procedures, and standards that govern the secure software development processes within your organization, focusing on the integration of security early on in the SDLC.

***Key stat: According to***[***Gitlab***](https://about.gitlab.com/developer-survey/#security)***, half of security professionals in 2023 report that developers fail to identify 75% of vulnerabilities.***

In this blog, we’ll cover some secure SDLC best practices and better [**workflows**](https://vulcan.io/blog/cyber-security-automation-when-to-automate-your-vulnerability-management-program/) for more [**secure coding**](https://vc.vulcan.io/rs/841-RSS-862/images/Vulcan-Cyber-Secure-coding-principles.pdf?_ga=2.103793024.1444632616.1701597411-67544556.1689580646) in 2024 and beyond.

What you’ll learn

1. [**Understanding secure SDLC**](https://vulcan.io/blog/secure-sdlc-best-practices/#understand)
2. [**The importance of cyber risk management for SDLC**](https://vulcan.io/blog/secure-sdlc-best-practices/#importance)
3. [**What are the benefits of a secure software development lifecycle?**](https://vulcan.io/blog/secure-sdlc-best-practices/#benefits)
4. [**Popular secure SDLC methodologies**](https://vulcan.io/blog/secure-sdlc-best-practices/#popular)
5. [**11 best practices for secure SDLC**](https://vulcan.io/blog/secure-sdlc-best-practices/#best)
6. [**How to get started with SDLC best practices**](https://vulcan.io/blog/secure-sdlc-best-practices/#start)
7. [**Secure coding practices and tools for better cyber risk management**](https://vulcan.io/blog/secure-sdlc-best-practices/#tools)
8. [**Next steps**](https://vulcan.io/blog/secure-sdlc-best-practices/#next-steps)

**Understanding secure SDLC**

**Secure SDLC is the practice of integrating security activities**, such as creating security and functional requirements, code reviews, security testing, architectural analysis, and risk assessment, into the existing development workflow. This might, for example, involve **writing your security and business requirements together** and performing a [**risk analysis**](https://vulcan.io/blog/risk-management-risk-assessment-risk-remediation-what-are-the-differences/) of your architecture during the design phase of SDLC.

***Key stat:***[**47%**](https://zipdo.co/statistics/software-development/)**of software development companies say that data breaches are the most significant risk when using a new technology.**

Throughout every stage of the secure software development lifecycle, security methods and [**remediation tools**](https://vulcan.io/resources/the-best-free-and-open-source-tools-for-cyber-risk-assessment-and-mitigation/) are typically integrated with code repositories to address any concerns or potential vulnerabilities as they emerge.

[**Register now: The CyberRisk Summit | Tuesday, June 18, 11:00AM – 1:00PM EDT**](https://events.ringcentral.com/events/cyberrisk-summit-june-2024?utm_source=Vulcan%20Blog&utm_campaign=Organic%20blogs)

**What are the phases of SDLC?**

The SDLC is a series of phases that begin with planning and end with maintenance. Each phase of the SDLC is critical to the success of the project, and it is important to follow this process of secure coding practices in order to ensure that the software meets the needs of the end-users and functions as expected. The phases of SDLC are:

1. Planning: Define project scope, resources, timelines, and budget.
2. Analysis: Gather and document software requirements.
3. Design: Create a software design and plan.
4. Development: Build and test the software.
5. Testing: Test the software for bugs and issues.
6. Deployment: Deploy the software to production.
7. Maintenance: Monitor, update, and maintain the software.

**The importance of cyber risk management for SDLC**

Cyber risk management is a strategic approach employed to detect, analyze, prioritize, and implement defensive measures against any cyber risks that pose a threat to the organization.

Benefits of adopting a cyber risk management strategy include:

* Helps to mitigate possible cyber attacks.
* Reduces the potential for financial loss due to cyber attacks (a common attack motive).
* Helps preserve the organization’s reputation.
* Enhances the security of the organization’s sensitive data.

**Secure SDLC Policy & Cyber risk management frameworks**

Discussed below are some of the standardized frameworks you can use to help ensure more effective cyber risk management processes.

**NIST CSF**

The [**National Institute of Standards and Technology Cybersecurity Framework**](https://www.nist.gov/cyberframework) (NIST CSF) is one of the most popular frameworks for cyber risk management, offering a valuable [**breakdown of security requirements**](https://vulcan.io/resources/mastering-nist-compliance-with-vulcan-cyber) and recommendations. It organizes the risk management process into five core functionalities, namely:

* Identify
* Protect
* Detect
* Respond
* Recover

**ISO/IEC 27001**

[**The ISO/IEC 27001**](https://www.iso.org/standard/27001) was initially created by the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC). It provides standards by which security information can be systematically protected. Its core objectives are **confidentiality**,**integrity**, access control, and code quality and **availability**.

**What are the benefits of a secure software development lifecycle?**

The key benefits of secure SDLC include:

* **Improved security:**Secure SDLC fosters a proactive approach toward security-related rules and regulations. It enhances the security of your applications and allows all stakeholders to be informed about security concerns.
* **Cost reduction:**Incorporating code reviews early on in the SDLC can reduce the cost of managing and resolving security-related vulnerabilities. In other words, you save money by detecting and resolving issues as soon as they occur.
* **Regulatory compliance:**Secure SDLC provides a secure environment that meets the demands of your business and strengthens safety, security, and [**compliance**](https://vulcan.io/basics/the-ultimate-guide-to-cyber-security-compliance/). It helps detect design flaws early in the SDLC process, reducing business risks in your organization.

***Key stat: By adopting Agile SDLC for software development,***[***64%***](https://radixweb.com/blog/agile-statistics)***of the companies witnessed increased capability to manage changing priorities efficiently.***

**Popular secure SDLC methodologies**

Secure SDLC methodologies fall into two categories of secure coding practices:**prescriptive and descriptive.** The prescriptive approach tells the user what they should do and when. “Descriptives,” on the other hand, are descriptions of the actions taken by other organizations.

**Microsoft Security Development Lifecycle**

The Microsoft Security Development Lifecycle (SDL) is a prescriptive approach that covers most security aspects and provides guidance to organizations on how to achieve more secure coding. It helps build software that is compliant with regulatory standards, while lowering costs.

[**Read report: Vulcan Cyber named a leader by Forrester for Vulnerability Risk Management**](https://vulcan.io/resources/forrester-wave-vulnerability-risk-management)

**OWASP Software Assurance Maturity Model (SAMM)**

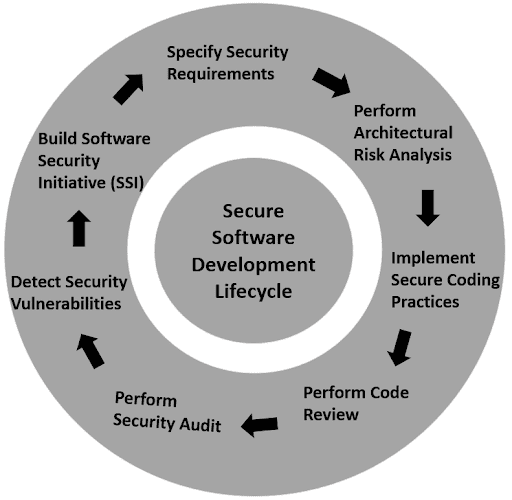
SAMM is an open-source project that follows a prescriptive methodology and guides the integration of security within the SDLC. OWASP maintains it, with contributions from companies of diverse sizes and industries.

**Building Security in Maturity Model (BSIMM)**

Initially a branch of SAMM, BSIMM has shifted from a prescriptive to a descriptive approach and is continuously updated with the most current best practices. Instead of advising about what actions to take, BSIMM summarizes the activities of member organizations.

**10 best practices for secure SDLC**

Securing your SDLC process requires embedding security into all phases of SDLC and adhering to the best practices outlined in this section.



**1. Specify your requirements**

Your specifications, security guidelines, and recommendations should be presented in a way that is simple and easy to understand in order to assist your developers.

**2. Perform security audits**

Security testing is critical for determining your product’s vulnerability to attacks. Ensure security tools and practices are in place from the outset and throughout the development process. It is also important to clearly define the functional requirements of your development teams, take into consideration common security vulnerabilities, and plan accordingly.

**3. Educate your developers on best coding practices, tools, and frameworks**

Organizing training sessions for your development team can help foster a **culture of security awareness**. These sessions should cover areas such as secure coding techniques, coding best practices, cyber security, potential risks, and the available security frameworks.

**4. Conduct an architectural risk analysis at the beginning**

Conducting an **architectural risk analysis** to identify flaws and determine risks that might occur due to those flaws should be done early on in the SDLC process, before coding your application. You should also take advantage of threat modeling to detect and manage threats in a timely manner.

**5. Tackle the big problems first**

Instead of addressing every vulnerability identified, it is important to [**prioritize the risks based on your business needs**](https://vulcan.io/vulnerability-prioritization-for-efficient-cyber-risk-mitigation/) and to concentrate on the most serious threats and feasible remedies. A triage approach—find, prioritize, and fix—can help you focus on avoiding security risks from existing vulnerabilities entering production environments and triaging and addressing compromised software over time.

**6. Secure planning and building for test cases**

Conducting a code review is important for verifying whether your development team has adhered to secure coding standards and practices, and allows you to uncover coding and configuration defects or weaknesses in the application. Make your plans ready for penetration testing on your application, and ensure that it is conducted by a third party. Organizations often employ third-party vendors to perform penetration testing. The primary objective of having a third-party vendor assess the security of your systems is to get an i**mpartial, professional, and expert opinion** on your security posture.

**7. Use code-scanning tools**

There are two types of [**analysis tools**](https://vulcan.io/blog/security-testing-101/): static analysis security tools **(SAST)** and dynamic analysis security tools **(DAST)**. While SAST enables you to analyze your code to identify security flaws in the application without running it, DAST is capable of finding flaws in your infrastructure.

**8. Cultivate a growth mindset**

Secure SDLC changes how teams operate and communicate. Everyone on the team should be open to learning, and your developers should be encouraged to adhere to the guidelines and best practices to secure the applications they build.

**9. Keep an eye on open-source security**

Open-source components with known vulnerabilities are another important factor to take into consideration when building a secure SDLC. Because **today’s software products rely heavily on open-source code**, it is critical to focus on open-source security management throughout the SDLC process.

Automated software composition analysis (SCA) tools can help determine security vulnerabilities in code and provide remediation insights and automatic patches.

10. Perform a gap analysis

A security gap analysis is a great way to check the integrity of your application. Performing a gap analysis will help you assess how effectively your system is operating based on your expectations. If there is any deviation from these expectations, you can identify which part of your SDLC needs to be re-examined in order to make the necessary improvements.

11. Create a software security initiative (SSI)

A software security initiative (SSI) is a process that allows you to plan for risk and allocate resources accordingly. An SSI guides you through the SDLC based on procedures and guidelines and helps you determine **how much you should spend on application security**. It also helps to ensure your team really understands their roles.

**The benefits of going cloud-native**

In addition to the best practices discussed above, being [**cloud-native**](https://vulcan.io/cloud-security-programs/) offers numerous benefits when it comes to security considerations and ensuring a secure SDLC. It enables:

* Building more reliable systems
* Faster deployments
* Avoiding vendor lock-in
* On-demand infrastructure
* Improved automation and collaboration
* Better customer experience
* Reduced costs through [**containerization**](https://vulcan.io/blog/secure-your-whales/)

**How to get started with SDLC best practices**

A well-thought-out SDLC implementation should complement an organization’s existing software development process. There are several recommended steps to help software developers get started with some of the secure SDLC best practices.

**Security training and awareness**

Security training and awareness sessions are a good starting point and an important part of secure SDLC. The sessions should involve all project team members—the development, QA teams, and release and maintenance teams, for example.

Make sure the sessions are easy to follow, focusing on concepts such as secure design principles, encryption, and security issues. The training should also cover cybersecurity risks, risk impact, and risk management.

**Technologies & tools for secure SDLC**

There are a number of available tools to help you implement secure SDLC:

* Static application security testing (SAST) tools such as **[SonarCube](https://www.sonarqube.org/" \t "_blank)** and **[FortifySast](https://www.microfocus.com/en-us/cyberres/application-security/static-code-analyzer" \t "_blank)**
* Dynamic application security testing (DAST) tools such as **[FortifyDast](https://www.microfocus.com/en-us/cyberres/application-security/webinspect" \t "_blank)**
* Software composition analysis (SCA) tools such as [**Dependency Check**](https://owasp.org/www-project-dependency-check/) and [**Dependency Track**](https://owasp.org/www-project-dependency-track/) from OWASP

**Shift left for secure SDLC**

Secure SDLC is one example of the **“shift-left” strategy**, which includes security checks early on in the SDLC process. This allows software development teams to plan releases more accurately, making it easier to detect and fix problems that may impact the release schedule. It therefore helps keep releases on track, while maintaining secure coding standards

***Key stat:*Approximately**[**79%**](https://zipdo.co/statistics/software-development/)**of developers believe that DevOps is critical to software development projects.**

The **[DevSecOps strategy](https://vulcan.io/blog/why-were-still-seeing-unpatched-software/" \t "_blank)** is all about having the right security policies, practices, and technologies from the beginning of the DevOps pipeline and integrating them throughout the entire SDLC process. Securing your SDLC enables you to offer secure products and services to your customers while still meeting tight deadlines.

**The role of AI in secure SDLC**

The use of Artificial Intelligence (AI) in application security is a rapidly evolving field. AI technologies are being integrated into security tools to enhance threat detection, automate security testing, and improve response times. These systems can analyze vast amounts of data to identify patterns and anomalies that indicate potential security threats.

AI-driven tools are particularly effective in identifying new, sophisticated attacks that traditional security measures might miss. The adaptive nature of AI also allows for continuous learning and improvement of security protocols, making it a valuable asset in the dynamic landscape of cyber security. While [**there are notable risks inherent within AI**](https://vulcan.io/blog/ai-hallucinations-package-risk), its implementation into application security represents a significant leap forward in preemptively identifying and mitigating potential vulnerabilities and threats.

**Secure coding practices and tools for better cyber risk management**

When it comes to secure coding, there are some industry-tested methods and strategies that teams can employ. The following tools and best practices can help you better manage risks in your organization.

**Bug bounties**

[**Bug bounties**](https://vulcan.io/blog/what-happens-when-bug-bounties-dont-work/) are financial rewards offered by companies in order to motivate ethical hackers to evaluate the company’s source code and identify bugs that pose a threat to the organization. T**he higher the reward, the greater the competition**, and the sooner a fix can be made available.

**OWASP (Open Web Application Security Project) Top 10**

OWASP is an open community devoted to providing free resources and guides to help improve your security. It lists the [**top 10 web application security risks**](https://vulcan.io/blog/owasp-top-10-vulnerabilities-2022-what-we-learned/), providing guidance for fixing them.