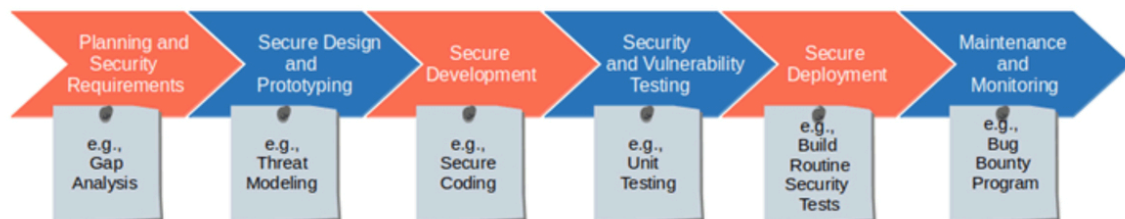


[1 INTRODUCTION TO SECURE SOFTWARE ENGINEERING](#) >[DIFFERENCES BETWEEN SDLC AND SSDLC](#)

# The Six Phases Secure SDLC

## Secure Software Development Life Cycle (SSDLC) Process



## The Six Phases in Secure SDLC

### 1. Planning and Security Requirements

In Secure SDLC, the **requirements phase extends beyond traditional functional specifications** by clearly defining security objectives and requirements. It involves detailed identification and documentation of security needs to ensure **comprehensive security coverage throughout development**.

#### Key Activities:

- **Security Requirements Analysis:**  
Involves examining business and system requirements from a security perspective. Security analysts and stakeholders collaborate to define clear and measurable security goals.
- **Misuse and Abuse Cases:**  
Development teams create misuse (unintentional misuse of system) and abuse cases (intentional misuse or malicious attack scenarios) to anticipate and prepare against potential security threats.

- **Non-Functional Security Requirements:**  
Explicit identification and documentation of essential security attributes, including confidentiality, integrity, availability, authentication, authorization, and auditability.
- **Requirements Traceability and Documentation:**  
Ensures all identified security requirements are well-documented, consistently tracked, and traceable through subsequent phases using tools such as protection profiles (PPs) and security targets (STs).

## 2. Secure Design and Prototyping

During the design phase, Secure SDLC emphasises embedding security principles deeply within the architecture of the application. Rather than retroactively addressing vulnerabilities, Secure SDLC proactively mitigates them by employing robust **secure design practices**.

### Key Activities:

- **Secure Architecture Design:**  
Architecture is designed with security as a primary objective, using layered defences (defense-in-depth), separation of duties, and minimal trust boundaries to limit potential attack surfaces.
- **Threat Modelling:**  
Systematic identification and evaluation of potential security threats using structured methodologies such as STRIDE (Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege), Attack Trees, and DREAD (Damage, Reproducibility, Exploitability, Affected Users, Discoverability).
- **Security Design Principles:**  
Employing security-centric design principles, including least privilege (limiting permissions strictly necessary for users or processes), fail-safe defaults, complete mediation (ensuring every access is validated), and secure default configurations.
- **Attack Surface Reduction:**  
Minimising components, code pathways, or services exposed to potential attackers, reducing the opportunities for exploitation.

### SSDLC – Phase Two – The Key Steps of the Threat Modeling Process



The graphic shows an example of threat modelling, a key process to include in your secure design and prototyping efforts.

Now that you’ve nailed the requirements, it’s time to put them “on paper” and describe how they should look once included in the application. Don’t forget to specify where and how security issues or concerns are addressed. In this step, you’ll ensure that:

- **All security requirements will be implemented and coded following the latest secure coding standards** (more on that in a minute). This means that the application will be created using the most up-to-date security architecture to protect it from the latest security threats.
- **The application will be built by applying secure design and threat modeling at every step of the secure SDLC.** This means that [threat modeling](#) (i.e., architectural risk analysis), [risks assessments](#), and security tests will become part of the software development process.
- **The application will be fully compliant with data privacy and security regulations.** Therefore, you’ll have to list, analyze, and describe how regulations and standards, like the following, will be addressed:
  - [General Data Protection Regulation](#) (GDPR),
  - [California Consumer Protection Act](#) (CCPA), and/or
  - [The Payment Card Industry Data Security Standard](#) (PCI DSS)

## 3. Secure Development

The implementation phase integrates security directly into the **coding process** by following **secure coding standards and practices**. This proactive approach mitigates vulnerabilities resulting from coding errors or insecure programming techniques.

## Key Activities:

- **Adherence to Secure Coding Standards:**

Following industry-established guidelines, such as [OWASP Secure Coding Guidelines ↗](#) or [CERT Secure Coding standards ↗](#), to prevent common vulnerabilities like;

- SQL injection,
- Cross-Site Scripting (XSS),
- Buffer Overflow, and
- Insecure object references

To learn more about the OWASP Top 10 vulnerabilities and how to mitigate them, visit the following [link ↗](#).

- **Static Code Analysis:**

Automated analysis tools (e.g., SonarQube, Fortify) are used to continuously scan source code, promptly identifying and fixing security issues and deviations from coding standards.

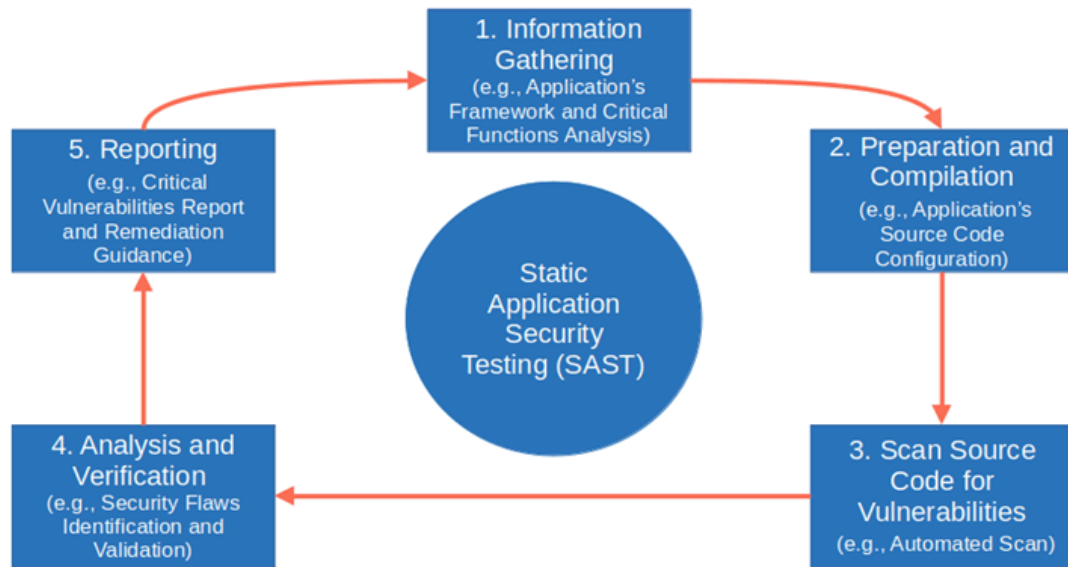
- **Code Reviews and Pair Programming:**

Conducting regular peer code reviews to ensure adherence to secure coding guidelines and leveraging pair programming practices to reduce errors and improve code quality.

- **Input Validation and Secure Logging:**

Implementing rigorous input validation techniques and maintaining secure, detailed logging to facilitate auditing and forensics without compromising sensitive data.

## SSDLC – Phase Three – How Static Application Security Testing (SAST) Works



Static Application Security Testing (SAST)

## 4. Security and Vulnerability Testing

Security testing in Secure SDLC is conducted comprehensively and rigorously, addressing security risks through diverse testing techniques. It ensures vulnerabilities are proactively identified and resolved before deployment.

### Key Activities:

- **Security Testing Techniques:**

Employing **Dynamic Application Security Testing (DAST)** to detect runtime vulnerabilities, **Static Application Security Testing (SAST)** for source code analysis, interactive application security testing (IAST) for runtime monitoring, and penetration testing to simulate real-world attacks.

- [DAST Tools ↗](#).

- **Risk-based Testing:**

Prioritization based on security risk profiles and focusing on high impact vulnerabilities, aligning testing resources towards areas of significant threat potential.

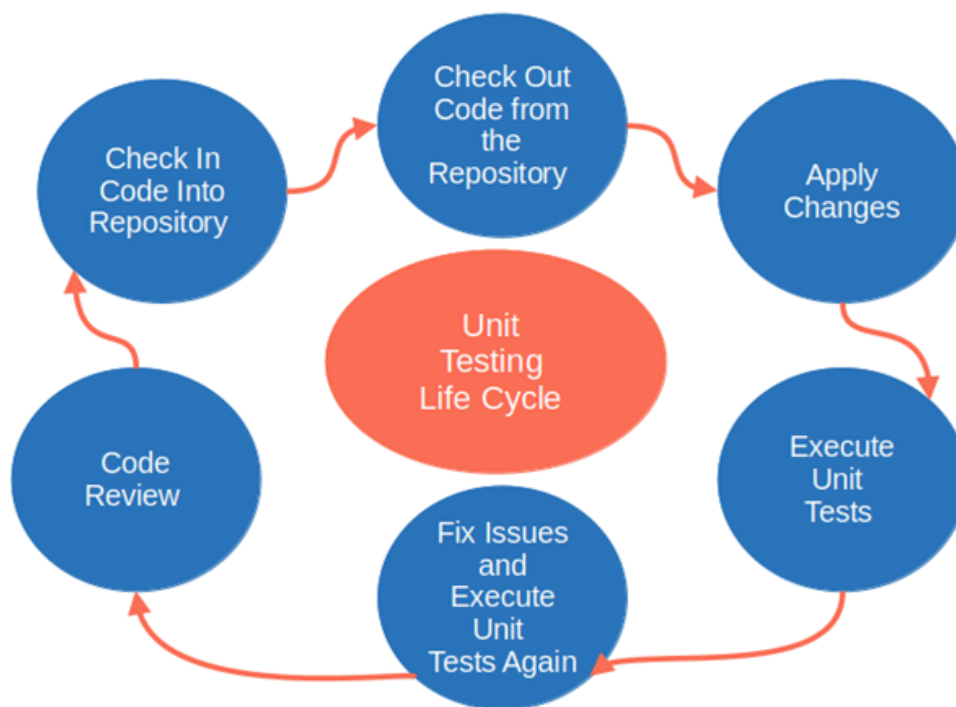


# Secure Software Engineering



- **White-box and Black-box Testing:**  
Applying comprehensive testing methodologies—white-box testing (internal system details accessible) and black-box testing (testing without internal knowledge)—to reveal security weaknesses from diverse attacker perspectives.
- **Evaluation Assurance Levels (EAL):**  
Implementing structured security assurance standards, such as those defined by the Common Criteria, to evaluate and demonstrate the robustness and reliability of security controls.

### SSDLC – Phase Four – Unit Testing Life Cycle



## 5. Secure Deployment

Secure deployment goes beyond simply releasing software into the operational environment, emphasizing secure configuration and validation to ensure that the system is securely set up and protected from potential exploitation from the outset.

### Key Activities:

- **Secure Configuration Management:**  
Ensuring systems are configured according to security best practices.
- **Post-deployment Validation:**  
Conducting security checks after deployment to ensure no new vulnerabilities were introduced during the transition from testing to the live environment.
- **Incident Response Preparation:**  
Developing clear and structured incident response plans to address potential security breaches rapidly.

## 6. Maintenance and Monitoring

Maintenance in SSDLC is a continuous process, proactively managing the software's security posture throughout its lifecycle, not merely correcting issues post-deployment.

Did you know that there was a [63% increase ↗](#) in white hackers reporting vulnerabilities in 2020? Yup, bringing white hackers on board means you can let the [others do the work ↗](#) for you. [Tap into their skills ↗](#), and compensate them if they find a bug or a flaw in your application. It'll save you money and time while helping you keep your application secure.

### Key Activities:

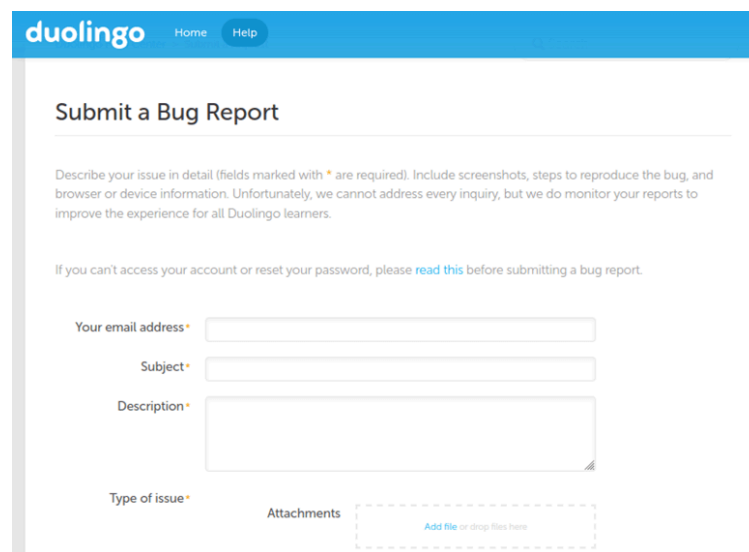
- **Continuous Security Monitoring:**  
Regular monitoring of the software for security incidents, anomalous behaviors, and vulnerabilities.
- **Patch Management:**  
Proactively applying security patches and updates to address vulnerabilities that arise over time.
- **Incident Response Management:**  
Developing and maintaining plans to respond promptly and effectively to any security incidents or breaches.
- **Periodic Security Assessments:**  
Regularly scheduled penetration tests, audits, and reviews to identify

emerging threats and vulnerabilities and continuously enhance the security stance.

- **Reactive and Preventive Security Measures:**

Deploying preventive security mechanisms and preparing reactive measures for swift incident response and mitigation.

Make it simple for users to report bugs with a 'Found a bug? Tell us!' button or a feedback page, like Duolingo or Facebook. Your users are key allies in spotting issues, for instance like Duolingo bugs report form.

The image shows a screenshot of the Duolingo website's 'Submit a Bug Report' form. The form is titled 'Submit a Bug Report' and includes instructions: 'Describe your issue in detail (fields marked with \* are required). Include screenshots, steps to reproduce the bug, and browser or device information. Unfortunately, we cannot address every inquiry, but we do monitor your reports to improve the experience for all Duolingo learners.' It also provides a link for users who can't access their account. The form fields are: 'Your email address \*', 'Subject \*', 'Description \*', 'Type of issue \*', and 'Attachments' (with a dashed box and the text 'Add file or drop files here').

Example of Bugs report form by Duolingo

## References

- **Secure Software Development Lifecycle SDLC:** <https://codesigningstore.com/secure-software-development-life-cycle-sdlc> ↗
- **Secure Coding Practices:** <https://owasp.org/www-project-secure-coding-practices-quick-reference-guide/stable-en/01-introduction/05-introduction> ↗

Previous  
Differences between SDLC and SSDLC

Next  
Security in Software Development



Last updated 2 months ago

---

