4. *Initial minimum requirements for testing software source code* including defining types of manual or automated testing (such as code review tools, static and dynamic analysis, software composition tools, and penetration testing), their recommended uses, best practices, and setting realistic expectations for security benefits.

Organizations that promote secure coding practices, perform rigorous testing, code audits and invest in developer education are more likely to ship more secure software. Consistent with the Shift-Left development philosophy, the software development life cycle should include supply chain management, asset inventory, an internal catalog of software composition, license compliance, and patch management.

By following secure coding practices, such as the SEI CERT secure coding standard together with checks against the OWASP top 10 and SANS CWE top 25 lists, both design flaws and implementation bugs during software development can be reduced. The Ericsson Security Reliability Model (SRM) ensures secure software development, delivery, implementation and maintenance across the product lifecycle. The SRM framework is enforced by utilizing risk assessment tools like static and dynamic code analysis, secure coding reviews, black-box and white-box vulnerability analysis and more. Tools are configured for the optimal coverage of these rules, and a manual review of the code enforces the rules that can't be verified with the tools. Code quality may have an impact on security and performance, so automated tools are also used also for tasks such as measuring code complexity, detecting the use of unsafe functions and finding duplicate code. Further information about Ericsson's best practice approach to product security and privacy by design are detailed in the following public document: https://www.ericsson.com/495435/assets/local/security/the-ericsson-security-reliability-model.pdf

Assurance refers to the activities conducted with the intention to ensure that the final product is secure when it is running in its target environment. Additionally, assurance work includes activities for continual improvement and adherence evaluation. Assurance activities can be grouped into five main categories:

- risk assessment

  A risk assessment uses systematic methodologies such as threat modeling with STRIDE to identify risks related to the product when used in the customer's network. Based on the risks identified and their evaluation, a risk mitigation plan is created. It mitigates the identified risks by introducing security controls or by suggesting other alternative means such as specific configuration to reduce the customer's risk exposure. One option to reduce a risk is to select an advanced level of security assurance activities to be performed during product development. A set of advanced and tailored activities are defined in the assurance areas: secure coding and vulnerability analysis. Examples of such activities are extended secure coding review and penetration testing performed by an external team.

- Privacy impact assessment

  A privacy impact assessment is a risk management activity that is integrated into the risk assessment workflow. The privacy impact assessment examines privacy threats based on identified assets and

their criticality for a given context. The assessment outputs residual privacy risks after treatment and forms the basis for implementation decisions that carry accountability. To identify privacy threats, apply the appropriate threat modelling methodologies (TRIM or LINDDUN).

- Secure coding

By following secure coding practices, both design flaws and implementation bugs can be reduced during the software development. The SRM follows the SEI CERT secure coding standard together with the OWASP top 10 and SANS CWE top 25 lists. SRM standards are enforced by tools like static and dynamic code analysis, and with secure coding reviews. Tools are configured for the optimal coverage of these rules, and a manual review of the code enforces the rules that can't be verified with the tools. Code quality may have an impact on security – for example, when the code is modified. Thus, tools are used also for tasks such as measuring code complexity, detecting the use of unsafe functions and finding duplicate code.

- Vulnerability analysis

A vulnerability analysis comprises the testing and verification of activities that are designed to identify weaknesses and vulnerabilities in the product or solution. The vulnerability analysis verifies the security characteristics and security configuration of the product/ solution and identifies new vulnerabilities through both black-box and white-box testing. Multiple tools and techniques are used, such as port scanning, vulnerability scanning, fuzzing and dynamic web application testing. Manual penetration testing is also performed. The verification of the security controls' functionality is done in the product's functional verification.

- Hardening

The term hardening refers to increasing the security of an application by reducing its attack surface. Hardening affects not only design, but also configuration and deployment. It ensures that the product is configured in a manner that minimizes the risk of unauthorized access and system compromise. Development-time hardening includes, for example, ensuring that processes are run with least privileges, removing unnecessary software components, updating to the latest patch level of the components, disabling insecure services and replacing default passwords.

Robust security assurance depends on appropriate activities conducted to ensure that the final product is secure when it is running in its target environment, with the right risk considerations. These activities include risk assessments, privacy impact assessments, secure coding, vulnerability analysis and hardening. Ericsson's Security Reliability Model (SRM) specifies the relevant assurance activities for each category in every phase of the product value flow: Source, Develop and Deliver. A continual improvement process for development and product lifecycle includes a root-cause analysis of the security flaws. The resulting improvements should be incorporated into the relevant design or processes.