Secure Software Development Policy

Users require trustworthy and secure software, and developers who address security threats more effectively than others can gain a competitive advantage in the marketplace. All software developers must address security threats using secure software development principles.

Purpose:

The purpose of this policy is to define the methodologies and principles within Wood to develop secure software.

The quality and integrity of Wood's applications must be designed and implemented in line with the applicable Wood security policies, procedures and standards using pre-defined application development principles and procedures.

Scope:

This policy is applicable to all internal development teams within Wood and to external development teams if developing applications for Wood.

Policy Requirements:

The application component of all information systems, whether it is developed in house or purchased from a third party, must be designed using secure software development principles. These principles must include security considerations and be applied to all stages of the Software Development Life cycle (SDLC).

Development of the application element of an information system must include the creation and execution of a security test and evaluation plan. The results of this test and evaluation must be documented and shared with key stakeholders.

Any external release of software that can be installed on a user's device (internal or external), regardless of operating system or platform, must comply with security and privacy policies as described in the Information Security Policy.

The goal is to minimize security-related vulnerabilities in design, code, and documentation and to detect and eliminate vulnerabilities as early as possible in the SDLC. These improvements reduce the number and severity of security vulnerabilities, enable the confidentiality, integrity and availability of data and improve the protection of users' privacy.

Methodology:

Applications must be developed according to set methodologies that enforce security:

- Development processes must use documented and repeatable standards and processes.
- Training must be provided for the software development team focusing on embedding security in the design, development and deployment of software. There should be a focus on sharing learning throughout the team, learning from the wider experience and constant improvement
- Quality management must include security assessments and should be performed throughout the development process.
- Applications must be developed and tested in their own dedicated environment before deployed to production environments.
- Security must be enforced throughout the application's service life including maintenance and decommissioning.

Principles:

Applications must be developed based on the following principles:

Secure by Design

- **Secure architecture, design, and structure** – Developers must consider security issues as part of the basic architectural design of software development. Developers must review detailed designs for possible security issues, and design and develop mitigations for the most prevalent and impactful threats.
- **Threat modelling and mitigation** – Threat models must be created and utilized within the SDLC, and threat mitigations must be outlined in all design and functional specifications. The STRIDE methodology should be used to support threat modelling.
- **Elimination of vulnerabilities** – No known security vulnerabilities should remain in the code after project reviews without sufficient mitigations in place to remove the impact of the vulnerability and approvals to accept the mitigation. Where

appropriate, these reviews must include the use of analysis and testing tools to eliminate vulnerabilities.

- **Improvements in security** – Less secure legacy protocols, as identified during internal audit and governance, must be deprecated, and, where possible, users must be provided with secure alternatives that are consistent with industry standards.
- **Data protection by design** – Developers must consider privacy and data protection issues during the design phase and throughout the SDLC.

Secure by Default

- **From the outset** – Security should be considered in the initial design and not incorporated later.
- **Least privilege** – All components within the application must run with the fewest possible permissions.
- **Defence in depth** – Components within the application should not rely on a single threat mitigation solution that leaves users exposed if it fails.
- **Conservative default settings** – The development team must be aware of the attack surface for the product and minimize it in the default configuration.
- **Avoidance of risky default changes** – Applications must not make any default changes to the operating system or security settings that reduce security for the host environment. For example: applications that either open up firewall ports without informing the user or instruct users to do so without consideration of possible risks.
- **Data protection by default –** Developers must ensure that only minimal data is collected and processed, exclusively for the purpose for which it was collected and for the time that it is needed.
- **Treat the cause –** Security should fix the root of the problem and not the symptoms.
- **Evolution –** Security should evolve over lifetime of the solution to meet new threats.

Secure in Deployment

- **Deployment guides** – Prescriptive deployment guides must be published to outline how to deploy each feature of a program securely, including providing users with information that enables them to assess the security risk of activating non-default options (and thereby increasing the attack surface).
- **Security management** – Software administrators must have the ability to manage the ongoing security within the application with either embedded security analysis and management tools or the utilisation of existing tool sets.
- **Patch deployment tools** – Deployment tools must be provided to aid in patch deployment.

Communications

- **Security response** – Development teams must respond promptly to reports of security vulnerabilities and communicate information about security updates.

Definitions:

- **SDLC** (Software Development Life Cycle) – a process for planning, creating, testing, deploying and maintaining high-quality software in a systematic and cost-effective way.

Reference Documents:

- Information Security Policy, GIT-POL-110003
- Data Protection Policy, COP-PLD-100007
- Secure Software Development Standard, GIT-STD-110008

Name	Daniel Thomas
Position	Technical Excellence Senior Manager
Date	30 April 2024