Secure Coding Policy

1 Intro

A. Document owner

Security Team

Version 1.0 – 11/2024

B. Who is this document intended for?

Developers, testers

C. Summary

This Secure Coding Policy outlines the principles and guidelines that govern the overall approach to security during the Implementation and Testing phases of the (Secure) Software Development Life Cycle ((S)SDLC) within the organization. The secure principles for environment set-up, coding itself and different testing are outlined in this policy. By adhering to this policy, the product's vulnerabilities, security flaws and risks should be minimized.

2 Content

A. Objectives

- Minimize security vulnerabilities and flaws both in the development environment and the code itself
- Minimize the cost of fixes of security flaws
- Mitigate risks and data leaks
- Drive continuous improvement
- Data Protection
- Quality assurance

B. Scope

This policy applies to the implementation and testing phases of the SDLC and to everyone involved in the process. It considers the environment setup, coding itself, code review as well as testing.

The policy is based on the NIST: Secure Software Development Framework (SSDF) Version 1.1, Microsoft Security Development Lifecycle and OWASP Secure Coding Principles. The practices

used from each of the frameworks are clearly noted. Only the practices that could be incorporated into the process were used; however, the whole SSDLC has the potential for continuous improvement, and the policy can be extended for other practices in the future as well.

C. Roles and responsibilities

- Chief Information Security Officer (CISO):
  - Oversees the overall security strategy and ensures alignment with business objectives.
  - Approves security policies and procedures, including the ones for secure development.
  - Points out possible security implications during the demonstration of new changes if not consulted earlier.
- Security Team
  - Writes security policies (together with legal help if necessary).
  - Helps with risk analyses and threat modelling.
  - Regularly reviews and updates the policies, frameworks, etc.
  - Performs vulnerability scanning.
- Developers
  - Follow the security policies and secure coding principles.
  - Conduct threat modelling.
  - Keep updated Software Bill of Materials (SBOM).
  - Integrate security into pipelines and test security there as well.
- Product Management
  - Conducts analysis with regard to security as well.
  - Performs risk analysis of new changes.
- Quality Assurance team
  - Test the product also with security implications in mind.

D. Implementation phase

2.D.1 Production Environment

2.D.1.1 PO.1.3

Before using a third-party component, its security needs to be analysed. This can be done by searching for previous vulnerabilities, data leaks caused by the component, etc. If there is a record of such occasions, the response to that needs to be considered in order to decide if the component can be used – namely, the time to patch the vulnerability, the publisher's transparency, etc. If the response is not up to the company standards, the component cannot be used in the system.

2.D.1.2 PO.3

If a tool is used in any toolchain (web, security, etc.), the same rules for its choosing as for a component in 2.D.1.1 apply. It also needs to be regularly updated, upgraded or replaced if a new vulnerability is found. The tools' logs need to be monitored, and an audit trail should be available for each one.

2.D.1.3 PO.5

Each environment used in the SDLC needs to be separated, and access to the environments needs to be limited. Where possible, MFA and network segmentation must be used. The use of privileged access to any environment must be logged and monitored. The development environment also needs to be monitored for attempted or real attacks.

For the development of the products, only company-managed devices should be used. If there is a need to use another device, refer to the internal Bring Your Own Device (BYOD) policy and inform the security team of this instance. The internal IT team manages the devices and sets the necessary endpoint security that other employees cannot change.

2.D.1.4 PW.6

All compiler, interpreter, and build tools need to be updated to the newest stable version as soon as possible and chosen similarly to 2.D.1.1.

The chosen compiler should warn the developer of unsecured code during compilation. The building process should end with zero errors and possibly zero warnings as well. Each such warning should be analysed and possibly eliminated. The build environment has to be secured, with limited access and updated regularly.

The configuration should be regularly tested and approved, if possible.

2.D.1.5 MSDL 2

Where possible, the usage of Managed Identities instead of SAS tokens should be considered. The usage of secrets in the code itself must be limited. However, when the usage of secrets in the code is necessary, it must never be in plaintext, and secure credential storage should also be used. Just-in-time and least privilege access to the workload should be used where possible. The number of users with privileged access needs to be as small as possible and well-documented.

When AI is being used in the product, be it for development or in the product itself, the EU's AI Act and internal security rules written in the internal Security Policy apply.

2.D.1.6 MSDL 6

The goal should be to strive for a Zero Trust model – each access request should be treated as if it came from an untrusted network. The commits of new changes to the production branch also need to go through code review, and pull requests should always be approved.

2.D.2 Implementation

2.D.2.1 PS.1.1

The code is stored only in the Git repository on Azure DevOps. Other storage is forbidden, especially the developers' devices or any removable media. Only permitted personnel, mainly the developers or QA personnel, if needed, are permitted to access the code.

All changes to the code are tracked, and the person who made the change is the one responsible for it.

The code owner should have an overview of all the changes to the product and approve the merge requests. They do not need to approve every single change, as the code review process is used for that.

2.D.2.2 PW.4

Where possible, the developers must use already existing well-secured third-party components according to 2.D.1.1. If that is not possible, well-secured software components that comply with the SSDLC processes should be created in-house. However, this can often lead to less secure products, so it should be limited.

A list of forbidden third-party software components needs to be updated regularly and stored in a place where all developers can access it.

A complete Software Bill of Materials (SBOM) should also be kept up-to-date and available to every developer. The details about the supplier, the current version of the component in the product, licensing, dependencies and security status with known flaws and their fix must be included with every component in the SBOM.

2.D.2.3 PW.5

The coding itself is to be done in accordance with the OWASP Secure Coding Principles. Furthermore, the known vulnerabilities of the development languages and environment used should be checked and accounted for. Each developer is responsible for reading their code again and correcting security flaws before sending it for code review.

2.D.2.4 MSDL 5

If a dependency on open-source software (OSS) exists, it should be inventoried together with possible vulnerabilities. The OSS needs to be included in the SBOM.

2.D.3 Code review

2.D.3.1 PW.7

A code review is to be performed after the implementation of every new change and when the change is returned for correction. Furthermore, each change in code should be scanned for issues.

The code review should be done by a developer who was not involved in its design or implementation. Furthermore, a junior developer should not perform a code review for another junior developer. If that is needed, an additional code review by a more experienced developer needs to be performed.

All feedback from the code review needs to be marked visibly in the code or as a comment in the ticket and incorporated by the original developer.

The code review should be done based on the recommendations in the Definition of Done and with regard to the Secure Coding Policy.

2.D.3.2 MSDL 2

During the code review, the use of proven security features, languages, & frameworks according to 2.D.1.4 is to be checked. If the reviewer finds non-compliance, that is also a reason to return for correction.

E. Testing phase

The already done tests, such as unit testing and integration testing, should be used with every change. The configuration of operating environments should also be tested before the change is sent to production. The big versions of the product must also go through overall testing of at least the main functionalities right before being published.

The testers team must test the changes with security implications (which should be mentioned in the design of the change) in mind and test not only the functionality but also the security of the change.

2.E.1 PW.8

On top of all other testing, executable code testing should be considered and used where possible. This can be used as the last step to find previously unidentified vulnerabilities. The results of this, together with all other testing, must be documented in a place accessible by the developers. The root causes of issues need to be identified and also documented in order to limit them in the future.

2.E.2 RV.1.2 & RV.1.3

Automated code analysis with SAST tools needs to be performed regularly before each sprint ends. Furthermore, the SCA should also be performed. The internal Vulnerability Management Policy further describes the vulnerability management process.

2.E.3 MSDL 7

The product is also to be tested with a chosen dynamic analysis security testing (DAST) tool. For the time being, this is the Security team's responsibility, and the process for managing found vulnerabilities is further described in the internal Vulnerability Management Policy.

On top of the testing with SAST, DAST and SCA tools, penetration testing is to be performed regularly. A certified company must perform an External penetration test at least annually. Internal penetration testing with prior knowledge is also to be considered.