Codage

**1. Objet**

**1.1 Finalité**

**1.2 Domaine d'application**

**1.3 Glossaire**

**2. Responsabilité**

**3. Documents**

**4. Exigences de la norme ISO 27001 : 2022**

**5. Déroulement**

**5.1 Planification**

**5.2 Codage**

**5.3 Revue et maintenance**

Historique

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
| Toutes | Création | 01/01/2022 |
| **Page** | **Changement** | **Date** |

**1. Objet**

**1.1 Finalité**

La présente procédure a pour finalité d'assurer que le logiciel est écrit en toute sécurité, réduisant ainsi le nombre de vulnérabilités potentielles de sécurité de l'information dans le logiciel.

**1.2 Domaine d'application**

Le domaine d’application de la procédure « Codage » s’applique aux personnes responsables du développement des logiciels.

**1.3 Glossaire**

SI – sécurité de l’information

RSI – responsable sécurité de l’information

IDE – *integrated development environment* - environnement de développement intégré (EDI)

SAST – *static application security testing* – test de sécurité des applications statiques

**2. Responsabilité**

Le responsable sécurité de l’information (RSI) a l’autorité de l’écriture et de la mise à jour de cette procédure. Le RSI est garant de son application. Il reçoit l’appui du directeur.

**3. Documents**

Codage sécurisé

Plan de test

Vulnérabilités techniques

**4. Exigences de la norme ISO 27001 : 2022**

A.8.28 Codage

Des principes de codage sécurisé doivent être appliqués au développement des logiciels.

**5 Déroulement**

**5.1 Planification**

Une base de référence minimale sécurisée est établie et appliquée afin d’assurer une bonne gouvernance du codage sécurisé, y compris pour les composants logiciels de tiers et les logiciels open source.

Les menaces du monde réel évoluent rapidement et sont surveillées étroitement. Des conseils et des informations à jour sont fournis sur les vulnérabilités logicielles afin de guider les principes de codage sécurisé grâce à un apprentissage continu.

Les principes de codage sécurisé sont utilisés à la fois pour les nouveaux développements et dans les scénarios de réutilisation. Ces principes sont appliqués aux activités de développement à la fois au sein de l'organisation et pour les produits et services fournis par l'organisation à d'autres. La planification et les conditions préalables avant le codage incluent les éléments suivants :

* les attentes spécifiques à l'organisation
* les principes approuvés pour le codage sécurisé à utiliser pour les développements de code internes et externes
* les pratiques de codage courantes et historiques
* les défauts qui conduisent à des vulnérabilités de sécurité de l'information
* les outils de développement utilisés comme des environnements de développement intégrés (EDI)
* le suivi des conseils des fournisseurs d'outils de développement et d'environnements d'exécution
* la maintenance et l’utilisation d'outils de développement mis à jour (compilateurs)
* la qualification des développeurs en écriture de code sécurisé
* la conception et l’architecture sécurisées, y compris la modélisation des menaces
* la sécurisation des normes de codage et leur utilisation imposée
* l’utilisation d'environnements contrôlés pour le développement

**5.2 Codage**

Les considérations lors du codage incluent les éléments suivants :

* des pratiques de codage sécurisées spécifiques aux langages et techniques de programmation utilisés, cf. le fichier Codage sécurisé
* l’utilisation de techniques de programmation sécurisées (programmation en binôme, l'examen par les pairs, les itérations de sécurité et le développement piloté par les tests)
* l’utilisation de techniques de programmation structurée
* la documentation du code et la suppression des défauts de programmation (exploiter les vulnérabilités de la sécurité de l'information)
* l’interdiction d’utiliser des techniques de conception non sécurisées (mots de passe codés en dur, échantillons de code non approuvés et services Web non authentifiés)

Les tests sont effectués pendant et après le développement, cf. le Plan de test.

Les processus de test de sécurité des applications statiques (SAST) peuvent identifier les vulnérabilités de sécurité dans les logiciels.

Avant de rendre le logiciel opérationnel, les éléments suivants sont évalués :

* surface d'attaque et principe du moindre privilège
* effectuer une analyse des erreurs de programmation les plus courantes et documenter que celles-ci ont été atténuées

**5.3 Revue et maintenance**

Une fois le code rendu opérationnel les éléments suivants sont pris en compte :

* les mises à jour sont empaquetées et déployées en toute sécurité
* les vulnérabilités de sécurité de l'information signalées sont traitées selon la politique Vulnérabilités techniques
* les erreurs et les attaques présumées sont enregistrées et les journaux régulièrement examinés afin d’apporter des ajustements au code, si nécessaire
* le code source est protégé contre l'accès non autorisé et la falsification (en utilisant des outils de gestion de la configuration, qui fournissent généralement des fonctionnalités telles que le contrôle d'accès et le contrôle de la version)

Quand des outils et des bibliothèques externes sont utilisés les éléments suivants sont pris en compte :

* les bibliothèques externes sont gérées (en maintenant un inventaire des bibliothèques utilisées et de leurs versions) et régulièrement mises à jour avec les cycles de publication
* la sélection, l'autorisation et la réutilisation de composants sont bien contrôlés, en particulier les composants d'authentification et cryptographiques
* la licence, la sécurité et l'historique des composants externes sont surveillés
* le logiciel est maintenable, suivi et provient de sources éprouvées et fiables
* une disponibilité suffisamment longue à long terme des ressources et des artefacts de développement est assurée

Lorsqu'un progiciel est modifié, les éléments suivants sont pris en compte :

* le risque que les contrôles intégrés et les processus d'intégrité soient compromis
* si le consentement du vendeur doit être obtenu
* la possibilité d'obtenir les modifications requises du fournisseur sous forme de mises à jour standard du programme
* l'impact si l'organisation devient responsable de la maintenance future du logiciel à la suite de modifications
* la compatibilité avec d'autres logiciels en cours d'utilisation

Un principe directeur est de s'assurer que le code pertinent pour la sécurité est mis en question et qu'il est inviolable.

Le code interprété peut être exécuté sur un service en nuage où l'accès au code lui-même nécessite des privilèges d'administrateur. Cet accès administrateur est protégé par des mécanismes de sécurité tels que les principes d'administration juste à temps et une authentification forte. Si le propriétaire de l'application peut accéder aux scripts par un accès distant direct au serveur, un attaquant le peut aussi, en principe.

Le code d'application est mieux conçu en partant du principe qu'il est toujours sujet à des attaques, par erreur ou action malveillante. De plus, les applications critiques peuvent être conçues pour être tolérantes aux défauts internes. Par exemple, la sortie d'un algorithme complexe peut être vérifiée pour s'assurer qu'elle se situe dans des limites sûres avant que les données ne soient utilisées dans une application telle qu'une application critique pour la sécurité ou la finance. Le code qui effectue les vérifications aux limites est simple et donc beaucoup plus facile à prouver.