code-maturity-assessor

code-maturity-assessor

> *Catalogue généré le 2026-05-11*

En une phrase

Évalue le "niveau de maturité" d'un codebase blockchain selon une grille Trail of Bits à 9 catégories (arithmétique, monitoring, contrôles d'accès, complexité, décentralisation, doc, MEV, low-level, tests) et te livre une note `Missing / Weak / Moderate / Satisfactory / Strong` par catégorie avec recommandations.

Quand l'utiliser

- Pour faire un **état des lieux objectif** d'un projet blockchain — pas un bug hunt, mais une note globale.
- Quand un investisseur, un partenaire ou toi-même veut savoir si un protocole est "prod-ready" ou encore en alpha.
- Avant un audit, pour identifier les chantiers structurels (= au-delà des bugs : est-ce que la gouvernance est trop centralisée ? Est-ce que la doc existe ?).
- Quand tu hésites entre 2 projets/forks et que tu veux comparer leur sérieux technique.
- Pour bâtir une roadmap d'amélioration priorisée.

Comment l'invoquer

- **Slash command** : `/code-maturity-assessor`
- **Phrases déclencheurs (texte)** : "assess code maturity" / "maturity scorecard" / "evaluate codebase quality"

- **Auto-invocation** : Sur demande explicite.

Description détaillée

Trail of Bits a formalisé une grille de "Code Maturity Evaluation v0.1.0" qui regarde un projet sous 9 angles, et c'est ce que ce skill applique de façon systématique. Contrairement aux scanners (qui cherchent des bugs précis), ici on évalue des **pratiques** : "est-ce qu'il y a des events bien définis pour le monitoring ?", "est-ce que les rôles admin sont bien séparés ?", "est-ce que la doc inline existe ?", etc.

Les 9 catégories : 1) **Arithmétique** (protection contre les overflows, gestion de la précision), 2) **Auditing** (events, monitoring, incident response), 3) **Authentification / Access Controls** (gestion des privilèges, multisig), 4) **Complexity Management** (taille des fonctions, profondeur d'héritage), 5) **Decentralization** (risques de centralisation, timelocks, opt-out utilisateur), 6) **Documentation** (specs, NatSpec, glossaire), 7) **Transaction Ordering Risks / MEV**, 8) **Low-Level Code** (assembleur, optimisations risquées), 9) **Testing** (coverage, fuzzing, tests d'invariants).

Workflow en 3 phases : **Discovery** (le skill scanne ton projet), **Analysis** (il vérifie chaque catégorie, te pose des questions sur les processus qu'il ne voit pas dans le code), **Report** (une scorecard avec ratings + roadmap priorisée).

Pour aller plus loin

Pour la grille complète avec les seuils de notation détaillés, consulter le SKILL.md original à `/home/thymon/.claude/plugins/cache/trailofbits/building-secure-contracts/1.0.1/skills/code-maturity-assessor/SKILL.md` et la ressource `resources/ASSESSMENT_CRITERIA.md`.

Source

- **Plugin** : `trailofbits/building-secure-contracts`
- **Nom interne** : `code-maturity-assessor`
- **Fichier** : `/home/thymon/.claude/plugins/cache/trailofbits/building-secure-contracts/1.0.1/skills/code-maturity-assessor/SKILL.md`

Revision #2
Created 2026-05-11 21:19:34 UTC by thymon
Updated 2026-05-11 21:37:11 UTC by thymon